02-15-00

A

U.S. Express Mail Label No. EL531623619US Attorney Docket No. 230074-0223

jc688 U.S. PTO 02/14/00

FOLEY & LARDNER
2029 Century Park East - Suite 3500
Los Angeles, CA 90067-3021

jc678 U.S. PTO 09/503282 02/14/00

TO: Assistant Commissioner for Patents
Box Patent Application
Washington D.C. 20231

UTILITY PATENT APPLICATION TRANSMITTAL UNDER 37 C.F.R. 1.53(b)

Transmitted herewith for filing is the patent application of:

INVENTOR(S): Lee Huynh and Roger Brouwer

TITLE: PACKET PROCESSOR

In connection with this application, the following are enclosed:

09503282 . 021400

APPLICATION ELEMENTS:
X Specification (including Abstract): 54 pages
X Claims: 31
X Drawings: 6 Sheets
Power of Attorney By Assignee and Exclusion of Inventor Under Rule 37 C.F.R. § 3.71
_ Declaration Under 37 C.F.R. § 1.63
_ Unsigned
_ Executed (original or copy)
_ Copy from a prior application (37 CFR 1.63(d))
(relates to continuation/divisional boxes completed) - NOTE: Box below

ACCOMPANYING APPLICATION PARTS:
_ Assignment Papers (cover sheet & document(s))
_Revocation and Power of Attorney
_ Information Disclosure Statement(IDS) with PTO-1449.
_ Copies of IDS Citations
_ Preliminary Amendment
_ Check No. 12586 in the amount of $1128.00 for the filing fee.
X Certificate of Mailing By Express Mail
X Return Receipt Postcard (MPEP 503)
X Appendix - 33 pages
_ Small Entity Statement(s)
_ Statement filed in prior application, status still proper and desired.
_ Certified Copy of Priority Document(s) with Claim of Priority (if foreign priority is claimed).

CONTINUING APPLICATION, check appropriate box and supply the requisite information:
_ Continuation _ Divisional _ Continuation-in-part (CIP) of prior application Serial No. _.
_ Amend the specification by inserting before the first line the following sentence: --This application is a_ continuation, _ divisional or _ continuation-in-part of application Serial No. _, filed _.--

CORRESPONDENCE ADDRESS:
Please address all future correspondence to:
Ted R. Rittmaster at the Foley & Lardner address noted above.
Telephone: (310) 277-2223
Fax Number: (310) 557-8475

FEE CALCULATIONS: (Small entity fees indicated in parentheses.)

(1) For	(2) Number Filed	(3) Number Extra	(4) Rate	(5) Basic Fee $690. ($345.)
Total Claims	31 - 20 = 11	11	x $18 (x $9)	$198.00
Independent Claims	9 - 3 = 6	6	x $78 (x $39)	$468.00
Multiple Dependent Claims: 0			$260 ($130)	0.
Assignment Recording Fee per property			$40	0.
Surcharge Under 37 C.F.R. 1.16(e)			$130 ($65)	0.
			TOTAL FEE:	**$666.00**

This Application is being filed without Fee or Declaration.

Respectfully submitted,

[illegible]

Ted R. Rittmaster
Reg. No. 32,933

February 14, 2000
Date

09503282.021400

015.417486.1

Express Mail Label No. EL531623619USS

Docket No. 230074-0223

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Provisional Application of	Attorney Docket No. 230074-0223
Lee Huynh and Roger Brouwer	Group Art Unit: Unassigned
Serial No.: Unassigned	Examiner: Unassigned
Filed: Concurrently herewith	
For: PACKET PROCESSOR	

CERTIFICATE OF MAILING BY EXPRESS MAIL

Box Patent Application
Assistant Commissioner for Patents
Washington, D.C. 20231

Assistant Commissioner:

I hereby certify that the following paper(s) along with any attachments referred to or identified as being attached or enclosed are being deposited with the United States Postal Service as Express Mail (Express Mail Label No. EL531623619US) under 37 C.F.R. § 1.10 on the date of deposit shown below with sufficient postage and in an envelope addressed to: Box Patent Application, Assistant Commissioner for Patents, Washington D.C. 20231.

1. Utility Patent Application Transmittal Under 37 C.F.R. 1.53(b)
2. Specification - 54 Pages
3. Drawings - 6 Pages
4. Appendix - 33 pages
5. Return Postcard

Respectfully submitted,

[illegible]

Ted R. Rittmaster
Reg. No. 32,933

February 14, 2000
Date

Foley & Lardner
2029 Century Park East, 35th Floor
Los Angeles, CA 90067-3021
Telephone: 310-277-2223
Facsimile: 310-557-8475

09503282 .021400

015.417640.1

VIA U.S. EXPRESS MAIL LABEL NO. EL531623619US

Attorney Docket: 230074.0223

PATENT APPLICATION IN THE U.S. PATENT AND TRADEMARK OFFICE

for

PACKET PROCESSOR

By

Lee Huynh and Roger Brouwer

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates in general to computer-implemented systems for processing packets, and, in particular, to performing encryption and authentication in parallel while processing packets.

2. Description of Related Art

In the last decade, the use of computers in both the home and office have become widespread. These computers provide a high level of functionality to many people. Additionally, the computers are typically connected to other computers via a network, such as the Internet and the World Wide Web (also known as "WWW" or the "Web"). Therefore, users are transmitting computer data between computers with increasing frequency. However, the growing use of computers to transmit data has resulted in extensive unauthorized use and copying of data while the data is stored on the computer or being transmitted between computers, costing owners of the data substantial revenue.

Moreover, with the fast growing popularity of the Internet and the World Wide Web (also known as "WWW" or the "Web"), there is also a fast growing demand for improved security of data. One technique for protecting data is by encrypting the data prior to transmitting the data. Cryptography involves transforming data into an unreadable format that can be deciphered with a "key" and transformed into a readable format. Cryptography may be used to protect various types of information, such as e-mail messages and personal information (e.g., bank account numbers). Another technique for protecting data is authentication. Authentication involves determining whether the source of particular data is a valid source. Authentication may involve using passwords or user names. This information may be appended to the data being transmitted so that the receiving computer can authenticate the data as coming from an appropriate transmitting computer.

Various forms of cryptographic units or modules have been developed for providing encryption and/or decryption functions. In addition, various forms of authentication units or modules have been developed for providing authentication functions. A combination of cryptographic and authentication units may be used, for example, to append authentication information to data before or after encryption. This authenticated and encrypted data may then be transmitted from a first computer to a second computer. The second computer uses the authentication information to determine whether the source of the data is a valid source, and also decrypts the data for processing.

If encryption and authentication are implemented in a single unit, then one packet must be encrypted and authenticated before the next packet is encrypted and authenticated. This continues until each packet is processed. One problem with this technique is that each packet to be encrypted must wait until the previous packet was both encrypted and authenticated.

On the other hand, if encryption and authentication are implemented in separate, independent design units, then the second packet need only wait until the required individual resource (rather than the combined encryption/authentication unit) becomes available. For instance, the second packet need only wait until the first packet has been encrypted and passed to the authentication unit. Once the first packet is passed to the authentication unit, the second packet may be provided to the encryption unit and encryption of the second packet may begin while authentication of the first packet proceeds. Thus, the system need not wait until the first packet is completely encrypted and authenticated before beginning encryption of the second package. However, the use of independent encryption and authentication units can result in significantly more communication traffic and congestion on the internal or external communication buses, as data is transferred between memory and each of the independent units.

Such congestion can increase processing latency and reduce throughput. The congestion problem increases as the number of authentication units and encryption units increases.

There is a need in the art for more efficient processing of packets to provide security in data storage and transmission.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art described above, to improve throughput and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses a method, apparatus, and article of manufacture for a computer-implemented packet processor.

In accordance with one embodiment of the present invention, a packet processor comprises one encryption unit and two authentication units arranged and controlled such that a packet may be encrypted and authenticated (once or twice) in a single pass through the packet processor. In preferred embodiments, the packet processor is controlled to begin processing a second packet as soon as the required encryption or authentication unit becomes available, while other units may still be processing the first packet. When only one authentication is required, performance is improved even more by using the two authentication units to perform one authentication (for HMAC- key hashing).

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 is an exemplary packet processor and system according to an embodiment of the invention;

FIG. 2 is an exemplary packet processor and system according to another embodiment of the invention;

FIG. 3 is an exemplary packet processor and system according to a preferred embodiment of the invention;

FIG. 4 is a timing chart, illustrating the timing of the processing of multiple packets;

FIG. 5 is a preferred example configuration of the packet processor and system of Fig. 3;

FIG. 6 is a block diagram illustrating a packet and a mask;

FIG. 7 is a block diagram illustrating outbound packet processing; and

FIG. 8 is a block diagram illustrating inbound packet processing.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In the following description of the preferred embodiment, reference is made to the accompanying drawings which form a part hereof, and which is shown by way of illustration a specific embodiment in which the invention may be practiced. It is to be understood that other embodiments may be utilized as structural changes may be made without departing from the scope of the present invention.

Packet Processor

One embodiment of the present invention provides a packet processor for processing packets of data. In particular, the packet processor encrypts data and/or decrypts encrypted data and provides at least one, and preferably multiple levels of authentication. In preferred embodiments, encryption (or decryption) and authentication are implemented in separate units, such that one or more resources (encryption, decryption or authentication units) may become available to process a data packet, while one or more other resources are processing another data packet. As a result, the system need not wait until the first packet is completely processed, for example, completely encrypted and authenticated, before beginning processing, for example, encrypting, the second package. In further preferred embodiments, a data bus configuration is employed for minimizing communication traffic congestion problems, as data is transferred to and from the separate encryption, decryption and authentication units.

In accordance with one embodiment of the present invention, as shown in FIG. 3, a packet processor 204 comprises a control unit 106, one crypto unit 108 and two authentication units 110 and 112, arranged and controlled such that a packet may be encrypted and authenticated (once or twice) in a single pass through the packet processor. However, other embodiments may employ any suitable number of authentication units and crypto (encryption and/or decryption) units. In a preferred embodiment, the control unit 106 comprises a RAM-based controller. However, the control unit may be implemented using any suitable programmable processor or other the like.

The cryptography component 108 may comprise, for example, a "crypto core" having a data encryption standard (DES), which is a symmetric-key encryption technique. With a symmetric-key technique, the sender and receiver of a message share a single key. Alternatively, the cryptography component 108 may use other encryption standards, including, but not limited to, 3DES and RC4. Further embodiments may employ other suitable cryptographic devices and encryption standards, for encrypting and/or decrypting data. The authentication units 110 and 112 may comprise any suitable authentication devices, including, but not limited to "HMAC cores" (e.g., HMAC-SHA-1 or HMAC-MD5), which use standard authentication hashing techniques.

In the system shown in FIG. 1, the packet processor 104 is coupled to a computer or main processor 100, through a processor bus 102. The computer or main processor 100 may comprise any suitable computer or data processing device, including, but not limited to a personal computer PC, PowerPC (PPC), mainframe, or the like, which may include, inter alia, a processor device and random access memory (RAM), and may be coupled to one or more data storage devices (e.g., hard, floppy, and/or CD-ROM disk drives, etc.), data communications devices (e.g., modems, network interfaces, etc.), a monitor (e.g., CRT, LCD display, etc.), a mouse pointing device, and/or a keyboard. An example memory device 101 is shown in Fig. 1, coupled to the processor bus 102. The computer 100 may be connected to other devices such as read only memory (ROM), a video card, bus interface, printers, etc., through the processor bus 102. Those skilled in the art will recognize that any combination of the above components, or other components, peripherals, and devices, may be used with the computer 100.

In the Fig. 1 embodiment, the controller 106 and the crypto and authentication units 108, 110, and 112 are each coupled to the processor bus 102 by a respective link and are each capable of being a bus master, to take over the processor bus 102 and perform its own read or write transfers. These devices may also act as slave devices and may be accessed by the computer 100, through the processor bus 102. Because the crypto unit 108 and the authentication units 110 and 112 each use a separate bus interface or link to the processor bus 102, each of those units may be used independently of the other units and may be independently activated to become part of the packet processor. Thus, the packet processor of Fig. 1 may be configured to provide a single cryptographic function (encryption and/or decryption), by activating only the crypto unit 108. Alternatively, the packet processor of Fig. 1 may be configured to provide a single cryptographic function and a single authentication function, by activating the crypto unit 108 and one of the authentication units, such as unit 110. In addition, the packet processor of Fig. 1 may be configured to provide a cryptographic function and two authentication functions, by activating the crypto unit 108 and both of the authentication units 110 and 112. One skilled in the art will recognize that other variations of the Fig. 1 embodiment may employ other combinations of one or more cryptographic units and one or more authentication units, each of which are coupled to a processor bus for read or write transfer operations independent of the other cryptographic or authentication units.

In the embodiment shown in Fig. 1, information to be processed by the packet processor 104 is provided to each of the units 106, 108, 110 and/or 112 over the bus 102, as described below. Fig. 1 may represent an embodiment example, wherein information is provided in the form of one or more packets, for encryption and appending of one or more authorization codes. In such an embodiment, the processor 100 reads packets of information from the processor bus 102 and operates to process the packets to identify data fields to be encrypted. Information to be encrypted is stored in a memory device, such as memory 101, for example, under the control of the processor 100. If the crypto unit 108 is available (not presently encrypting information from another packet), the information to be encrypted is read into the crypto unit 108, over the processor bus 102, from the memory device, such as memory 101. The crypto unit 108 operates to encrypt the information and write the information into a memory device, such as memory 101, over the processor bus 102. Thereafter, the crypto unit 108 is available to encrypt information from another packet. Thus, the crypto unit 108 may begin processing the information from a second packet, without the need to wait for the encrypted information from the first packet to be processed by the authentication units 110 and 112.

The encrypted information written into the memory device is read into a first one of the authentication units 110, which appends an authentication code to the information and writes the information and appended authentication code into a memory device, such as memory 101. Thereafter, the authentication unit 110 is available to authenticate information from another packet. Thus, the authentication unit 110 may begin processing the information from a second packet, without the need to wait for the information from the first packet to be processed by the authentication unit 112.

The encrypted information and appended authentication code may be read into the second authentication unit 112 over the processor bus 102, from a memory device, such as memory 101. The second authentication unit 112 appends a second authentication code to the information and writes the resulting information and authentication codes into a memory device, such as memory 101, over the processor bus 102. The processor 100 may then access the stored information and authentication codes and form a resulting packet composed of encrypted information and two levels of authentication codes. The resulting packet may then be communicated from the processor 100, over the processor bus 102 to a load, user, host or the like, or may be stored in a memory device, such as memory 101, for later use.

The packet processor shown in Fig. 1 may also represent a decryption embodiment example, wherein information is provided in the form of one or more packets, for decryption and verification of authentication codes. In such an embodiment, the processor 100 reads packets of information from the processor bus 102 and operates to process the packets to identify data fields to be authenticated and decrypted. Information to be authenticated is stored in a memory device, such as memory 101, for example, under the control of the processor 100. If the authentication unit 112 is available (not presently verifying the authenticity of information from another packet), the information to be authenticated is read into the authentication unit 112, over the processor bus 102, from the memory device, such as memory 101. The authentication unit 112 operates to verify the authenticity of the information in accordance with well known authentication techniques, to provide first authenticated information. The first authenticated information is communicated over the processor bus 102 and stored in a memory device, such as memory 101. Thereafter, the authentication unit 112 is available to authenticate information from another packet.

The information to be authenticated is also read from the memory device into the authentication unit 110, which verifies the authenticity of the information in accordance with well known authentication techniques, to provide second authenticated information. The second authenticated information is communicated over the processor bus 102 and stored in a memory device, such as memory 101. Thereafter, the authentication unit 110 is available to authenticate information from another packet.

The information in the packet to be decrypted may be read into the crypto unit 108 over the processor bus 102, from a memory device, such as memory 101. The crypto unit 108 decrypts the cipher text from the packet and writes the resulting decrypted information into a memory device, such as memory 101, over the processor bus 102. The processor 100 may then access the stored information and may then communicate the decrypted, authenticated, packet information from the processor 100, over the processor bus 102, to a load, user, host or the like, or may store the information in a further memory device for later use. For inbound packets, other processing orders may be implemented, since there is no data-dependency in the data to be processed by each unit.

While each of the above variations of the Fig. 1 embodiment involve either encryption and appending authentication codes or decryption and authentication verification, in further preferred embodiments, the crypto unit 108 is capable of providing both encryption and decryption functions and is controlled by the processor 100 to provide one of those two functions for a given packet. Similarly, each authentication unit 110 and 112 is capable of providing both authentication code appending and authentication verification functions and is controlled by the processor 100 to provide one of those two functions for a given packet.

Because each of the processing units (crytpo unit 108 and the two authentication units 110 and 112) are individually coupled to the processor bus 102 by a respective link and are each capable of being a bus master, to take over the processor bus 102 and perform read or write transfers, the processor 110 can be controlled to process information from multiple packets at the same time. For example, once the crypto unit 108 completes an encryption process on information from a first packet and writes encrypted data into a memory device, such as memory 101, for further processing by the other processing units, the crypto unit is free to read in and encrypt information from a second packet. While the crypto unit 108 operates on the second packet information, the authentication unit 110 and/or 112 may be operating on the information from the first packet. Another advantage of the illustrated architecture is that authentication processing may be expedited, even when only one authentication is required. More particularly, the secret key authentication algorithm, for example HMAC, may be split into two processes, including an inner hashing and an outer hashing process. The inner hashing is processed in the first authentication unit 110, the result of which is passed to the second authentication unit 112 to perform the outer hash. This frees the first authentication unit 110 to begin processing the next packet.

Thus, the architecture shown in Fig. 1, wherein each processing unit is individually coupled to the processor bus 102 for read and write operations, allows the packet processor to process multiple packets simultaneously and allows processing of the next packet to begin as the resources (such as the crypto unit 108 in the above example) become available, rather than requiring the cryptographic and authentication processing of the first packet to be completed before beginning the processing of a second packet. However, the independent coupling each processing unit 108, 110 and 112 to the processor bus 102 and the multiple read and write operations that are required over the processor bus to process a packet, can result in a relatively large number of communications over the processor bus and, thus, a high level of communication traffic. Also, while embodiments described above employ the processor 100 to control the processing units and the access and storage of information in memory 101, in further embodiments, the processor 106 may be programmed to perform some or all of the functions performed by processor 100 noted above.

A packet processor 204 according to a further embodiment of the present invention is shown in Fig. 2 and is configured to address the above-discussed communication traffic problem. The illustration in Fig. 2 shows a processor 100, memory device 101, and processor bus 102, similar to the corresponding devices described above with respect to Fig. 1. The packet processor 204 in Fig. 2 includes a control unit 106, a crypto unit 108 and two authentication units 110 and 112, similar to correspondingly numbered elements discussed above with respect to Fig. 1. As with the embodiment of Fig. 1, further versions of the packet processor 204 may include any suitable combination of one or more crypto units and one or more authentication units. However, unlike the embodiment of Fig. 1, the packet processor shown in Fig. 2 includes a local bus 202. The control unit 106, the crypto unit 108 and the two authentication units 110 and 112 are each coupled to the local bus 202, with the control unit 106 being the only master on the local bus 202. In addition, the controller and at least the last (and, preferably, all) processing unit(s) in the chain are each coupled to the processor bus 102, with the control unit 106 capable of performing reading and writing operations on the bus 102. The crypto unit 108, authentication unit 110 and authentication unit 112 are coupled for communication in a daisy-chain arrangement, as represented by the daisy-chain connection 206.

Thus, in the Fig. 2 embodiment, only the control unit 106 is coupled to the processor bus 102 for both read and write operations. The processing units in the chain (the crypto unit 108 and the authentication units 110 and 112) are each coupled to receive control instructions and/or data from the control unit 106, over the local bus 202, as discussed below.

The control unit 106 uses its busmaster interface to read in the processing parameters and input packet data. The control unit 106 passes processing parameters to the crypto unit 108, the authentication unit 110 and/or the authentication unit 112, over the local bus 202. Depending upon whether the packet is inbound or outbound and the type of processing required, the control unit 106 passes the input packet information on to the crypto unit 108, the authentication unit 110 and/or the authentication unit 112, over the local bus 202, in the appropriate order. For outbound packets, where the input payload is plain text, the cipher text output from the crypto unit 108 is transferred to the authentication units 110 and 112 for authentication via the daisy chain bus. Additionally, the authentication value provided by the authentication unit 110 is made available to the authentication unit 112 for security packets requiring two authentications. Thus, with the Fig. 2 architecture, the input packet may be read once into the packet processor 204, and up to three cryptographic operations may be performed on the packet. The resulting output packet is written out to the destination, over the processor bus 102, by the processing units (108, 110 and 112), as output data is generated.

In accordance with a preferred embodiment, upon receiving a command from the processor 100 to process a packet, the control unit 106 reads in the commands (instructions) and parameters from the memory device 101 over the processor bus 102 and communicates information to the units that are required to process the packet (units 108, 110 and 112) over the local bus 202. In this manner, the authentication units are set up and ready to accept and process data as it arrives. In further preferred embodiments, the authentication units are capable of buffering instructions, parameters and data for the next packet, while processing an earlier packet.

More specifically, in a system in which one encryption and two authentications are performed on each packet as illustrated, the control unit 106 reads in packet data and passes data to the applicable unit, as follows. For outbound packets, the control unit 106 passes packet data to crypto unit 108, over the local bus 202. The crypto unit 108 encrypts the data and writes out the encrypted data to the memory device 101. The crypto unit 108 may also pass encrypted data to the authentication units 110 and 112 over the daisy chain connection 206. Authentication unit 110 processes the data and writes out authentication code to the memory device 101. Authentication unit 110 may also pass the authentication code to authentication unit 112 to be processed. Authentication unit 112 processes the data and writes out an authentication code to the memory device 101.

For inbound packets, the control unit 106 passes packet data to crypto unit 108, authentication unit 110 and authentication unit 112, over the local bus 202. The crypto unit 108 decrypts the data and writes decrypted data out to the memory device 101. Authentication unit 110 verifies the authentication code and sets a status bit based on the verification results. Authentication unit 112 verifies a further authentication code and sets a status bit based on the verification results. The control unit 106 collects the authentication verification status bits and writes status information to the memory device 101, over the processor bus 102.

Similar to the embodiment shown in Fig. 1, the bus architecture of Fig. 2 allows the ability to process multiple packets simultaneously and to use resources (for example, crypto and authentication units) as such resources become available, rather than waiting until each packet is fully processed before starting the processing of the next packet. However, the use of the local bus 202 and daisy chain bus 206 allows the control unit 106 and processing units 108, 110 and 112 in Fig. 2 to pass information therebetween, without using the processor bus 102. Thus, because the crypto unit 108 and the authentication units 110 and 112 do not use the processor bus 102 for reading input data and because intermediate data is not passed between those units over the bus 102, traffic on the processor bus 102 can be reduced or minimized. In addition, because the processing units are coupled to the processor bus 102 only for communication of the fully processed packet information, the communication traffic demand is minimized, relative to the Fig. 1 embodiment.

In yet further versions of the Fig. 2 embodiment, the fully processed packet information may be communicated from the second authentication unit 112 to the control unit 106, which then communicates the processed packet information over the processor bus 102. In such further versions, the connection between the second authentication unit and the processor bus 102 may be omitted.

. Similar to the Fig. 1 embodiment, another advantage of the illustrated architecture in Fig. 2 is that authentication processing may be expedited, even when only one authentication is required. More particularly, the secret key authentication algorithm, for example HMAC, may be split into two processes, including an inner hashing and an outer hashing process. The inner hashing is processed in the first authentication unit 110, the result of which is passed to the second authentication unit 112, for example, over the daisy chain bus 206, to perform the outer hash. This frees the first authentication unit 110 to begin processing the next packet.

A packet processor 304 according to a preferred embodiment of the present invention is shown in Fig. 3 and includes aspects of both of the embodiments discussed above and shown in Figs. 1 and 2. The packet processor 304 in Fig. 3 includes a control unit 106, a crypto unit 108 and two authentication units 110 and 112, similar to the packet processor shown in Figs. 1 and 2. As with the embodiments of Figs. 1 and 2, further versions of the packet processor 304 may include any suitable combination of one or more crypto units and one or more authentication units.

The packet processor 304 in Fig. 3 includes individual connections between each of the processing units (crypto unit 108, authentication units 110 and 112) and the processor bus 102, similar to the embodiment shown in Fig. 1, and a local bus 202 and daisy chain connection 206, similar to the embodiment shown in Fig. 2. Thus, the embodiment of Fig. 3 may operate in accordance with the above-discussed operation of the packet processor 104 in Fig. 1 or the operation of the packet processor 204 in Fig. 2.

The bus architecture described above allows for improved throughput and reduced traffic over the shared processor bus 102. When multiple cryptographic processing is required, data is passed from one unit to another in a somewhat pipelined manner. This implies that one unit may become available before the other units become available. Depending upon the cryptographic processing requirement of the arriving packets, the packet processor (104, 204 or 304) may start processing the next packet as soon as the required resources become available. Fig. 4 shows an example timing schedule for processing four packets, back-to-back, where all packets require the use of all three processing units. The first and third packets are outbound, while the second and fourth packets are inbound. For outbound packets, the authentication units 110 and 112 receive the cipher text from the crypto unit 108 and, thus, finish processing the packet after the crypto unit 108 has completed its operation. On the other hand, inbound packets already contain cipher text and, thus, the authentication units do not need to wait for data from the crypto unit 108. The second, third and fourth packets are started as soon as the crypto unit 106 finishes processing the previous packet.

One example configuration and operation of the packet processor and system of Fig. 3 is shown and described with respect to the packet processor 404 and system of Fig. 5. Another example configuration is shown and described in the Appendix attached hereto, which is incorporated herein by reference.

With reference to Fig. 5, the control unit 106 passes parameters from the packet control structures to the cryptography 108 and authentication 110 and 112 components via the packet processor local bus 126. The packet control structures indicate the particular techniques and parameters to be used to encrypt and authenticate a packet. The cryptography component 108 encrypts data received in a First-In-First-Out input queue (InFIFO) 128 and passes encrypted data to the authentication components 112 and 114 and to the First-In-First-Out output queue (OutFIFO) 130, to be written out to a destination such as the memory 101, over the bus 102. That is, the cryptography component 108 passes data to the InFIFO queue 132 of authentication component 110 and to the InFIFO queue 134 of the authentication component 112 via the daisy chain bus 136. In this manner the authentication components 110 and 112 receive the encrypted data for authentication. For a packet that is encrypted and authenticated for transmittal, the status unit 114 contains status information about whether the encryption and authentication processes completed successfully. For a received packet that is decrypted, the status unit 114 reports the status of the authentication value comparison along with the status of whether the decryption and authentication were processed successfully.

Those skilled in the art will recognize that the exemplary environment illustrated in FIG. 5 or in the Appendix is not intended to limit the invention. Indeed, those skilled in the art will recognize that other alternative hardware environments may be used without departing from the scope of the invention.

Typically, the data is sent in packets, which are units of data transmitted over a packet-switching network. Internet Protocol (IP) specifies the format of packets. IP security refers to a set of protocols being developed by the Internet Engineering Task Force, which is a standards organization for the Internet. The IP security protocols support transport encryption mode, in which only the data portion of each packet is encrypted, and tunnel encryption mode, which encrypts both the header and the data portion of each packet.

The packet processor 404 supports the Encapsulating Security Payload ESP and Authentication Header AH protocols as specified for IP security. Specifically, the following transformations are supported for AH protocols:

- SHA-1-96
- MD5-96
- HMAC-SHA-1-96
- HMAC-MD5-96

These are standard check sum functions using standard hashing techniques. For authentication, the packet processor 404 could use any of these techniques or some combination of them. For example, the H1 core 110 may use HMAC-SHA-1, while the H2 core 112 may use HMAC-MD5.

The following transformations are supported for ESP protocols:

- DES-CBC
- DES-CBC with HMAC-SHA-1-96
- DES-CBC with HMAC-MD5-96
- 3DES-CBC
- 3DES-CBC with HMAC-SHA-1-96
- 3DES-CBC with HMAC-MD5-96

The packet processor 404 will also support RC4, SHA- 1, HMAC-SHA- 1, MD5, and HMAC-MD5 standards.

The RAM-based control unit 106 has a small instruction set and a number of dedicated registers. Each instruction performs an operation or a group of operations. These instructions are used to form service routines to process data according to the desired protocol. The general purpose processor 100, such as a PowerPC (PPC), writes these routines to the Instruction RAM 116 via the processor local bus 102 slave interface of the control unit 106.

For each packet to be processed, a control structure is required to describe the packet (such as offsets and sizes), to pass parameters, and specify the cryptographic function to be performed. This can be done by the processor 100 or a host computer. The location of the packet control structure, packet data, and packet processing results are required for each packet to be processed. The Cmd FIFO (i.e., command "First in first out" queue) 118 provides a queue for storing pointers to control structures. For the packet processor 404, all data are aligned on a 32-bit word boundary.

Writing to a command register in the control unit 106 starts the packet processor 404. The packet processor 404 processes commands from the command FIFO 118 according to the mode selected in the command register. For each packet, the packet processor 404 retrieves the instruction RAM 116 offset and packet control structure location from the command FIFO 118 and starts executing the instructions beginning from the specified offset.

Data is first transferred into the control unit InFIFO 122 via the processor local bus 102 master read interface of the control unit 106. Data is processed as it becomes available in the InFIFO 122. The OutFIFO 124 is used to queue data that is to be copied to the destination memory 101, such as IP headers.

The Mask RAM 120 is preferably programmed with a mask pattern for masking out mutable fields in the header of the packet. The Mask RAM in the illustrated embodiment is 8 bits deep and 32 bits wide. Each bit is used to mask out one byte of data, and is applied to data beginning from the first byte in the packet. Thus, with the 8x32 Mask RAM, the first 256 bytes of the packet data may be masked out. The Mask RAM stores mask data for packet data that is in big endian format. If the packet data is in little endian format, then the control unit 106 swaps the bits.

FIG. 6 illustrates the format of the first 32-bit word 602 of the mask pattern. The leftmost nibble (bits 31-28) is to be applied to the first 32-bit word of the packet data, while the next nibble (bits 27-24) is to be applied to the next 32-bit word of the packet data, and so on. FIG. 6 also shows the mapping of the four bits of the leftmost nibble 604 to the first four bytes of the packet data. A mask bit of " 1" indicates that the data byte is to be masked to all zeroes. A mask bit of " 0" indicates that the data byte is not to be masked.

FIG. 7 is a block diagram illustrating outbound packet processing. A packet 700 initially comprises an IP header 702, an AH portion 704, an ESP header 706 and a payload (i.e., data) 708. For outbound packet processing, the payload 708 is encrypted with the cryptography core 108. Then, the H1 core 110 authenticates the encrypted payload 708 by adding an ESP authentication 710. Next, the H2 core 112 authenticates the encrypted packet 700 and the ESP authentication 710 by adding another authentication code (i.e., H2 result) 712.

FIG. 8 is a block diagram illustrating inbound packet processing. A packet 800 is received. The authentication added by the H2 core 112 is placed by the transmitting host into the AH portion 805 before processing the packet. Upon receiving the packet 800, the packet processor 404 performs authentication and compares the results against the ESP authentication

810 and the AH authentication information 805. The packet processor 404 also decrypts the encrypted data 808.

If the authentication is not valid, indicating that the packet may be invalid, the decrypted data is discarded. Although the packet processor 404 may be programmed to perform authentication verification first, and then perform decryption only if the authentication is valid, in preferred embodiments the packet processor 404 is programmed to perform authentication and decryption at the same time to minimize overall processing time and significantly improve throughput.

<u>Packet Processor Registers and Memories</u>

The following Table A indicates a representative example of the packet processor's 404 registers and memories. All offset and size registers are 16-bit wide and indicate the number of bytes. However, the packet processor 404 ignores the two least significant bits since everything has to be on a 32-bit boundary. All offset registers increment while executing the Write_data instruction.

<u>Table A</u>

CU Reg/mem	PLB R/W	PLB Addr [11:2]	PP Local Bus Addr (6:0]	Description
Offset Registers 0	r only	000	00	
Offset Registers I	r only	001	01	Crypto start offset
Offset Registers 2	r only	002	02	H1 start offset
Offset Registers 3	r only	003	03	H2 start offset
Size Registers 0	r only	004	04	Total packet data size
Size Registers I	r only	005	05	Crypto size
Size Registers 2	r only	006	06	H1 size
Size Registers 3	r only	007	07	H2 size
Crypto Command			08	Command formed from

				PCR, CSR, and Config Reg
H1 Command			09	Command formed from PCR, CSR, and Config Reg
H2 Command			0A	Command formed from PCR, CSR, and Config Reg
Status Destination Address Register	r only	00B	0B	Address for packet status
Offset Registers 4	r only	00C	0C	
Offset Registers 5	r only	00D	0D	Crypto end offset
Offset Registers 6	r only	00E	0E	H1 end offset
Offset Registers 7	r only	00F	0F	H2 end offset
Instruction RAM Pointer	r/w	010		Store offset for instruction RAM
Command/Status Register	r/w	011		Control packet processor (see bit definitions)
Packet Status	r only	012		Packet status
Source Address Register	r only	013		Starting address for packet control structure
Configuration Register	r/w	014		Control packet processor (see bit definitions)
Packet Command Register	r only	015	15	Select techniques and mode (see bit definitions)
Base Address Register	r only	016	16	Starting address for packet data (input)
Destination Address Register	r only	017	17	Starting address for packet results (output)
Mask RAM (0-7)	r/w	018-01F	18-1F	
InFIFO/OutFIFO	r/w	020-03F		
Command FIFO	r*/w	040		instruction RAM

				offset
Command FIFO	T*/w	041		pkt control structure address (load FIFO)
Instruction RAM (512 locations)	r/w	200-3FF		
Crypto Registers			20-213	
H1 Registers			40-4F	
H2 Registers			60-6F	

r* = reading command FIFO via PLB bus does not advance FIFO pointer

Packet Processor Instructions

The following Table B provides a representative example of packet processor 404 instructions.

Table B

Instructions	Description
• Load [#words]	• [#word] - 10 bits wide • _Load the specified number of 32-bit words into the Source Size Register and start transferring data into the CU InFIFO using the PLB Master Read interface until size reaches zero • The Source Address Register specifies the source address. • If the InDMA Enable bit of the command register is clear, no data transfer will occur.
• Storex [start address] [count]	• [start address] - 7 bits wide • [count] - 3 bits wide • Store [count + 1] words from the CU InFIFO (unmodified, except as mentioned below) to registers/memory beginning from the specified [start address]. • Destinations supported for this instruction are the following: • CU: Mask RAM, Base Address Register,

	Destination Address Register, Status Destination Address Register, Packet Command Register, Offset registers (0-7), and Size Registers (0-3). • Cores: All registers in the Crypto, Hl, and H2 cores. Need to execute the Wait instruction to make sure the cores are ready to accept data. • Storing size to Size Register I also cause (i) Crypto end offset to be computed and stored in Offset Register 5 and (ii) Crypto destination address to be computed and. stored in Crypto Destination Address Register. • Storing size to Size Register 2 also cause (i) H1 end offset to be computed and stored in Offset Register 6 and (ii) H1 destination address to be computed and stored in H1 Destination Address Register. • Storing size to Size Register 3 also cause (i) H2 end offset to be computed and stored in Offset Register 7 and (ii) H2 destination address to be computed and stored in H2 Destination Address Register. • This instruction can also be used to load commands (based on the information provided in the Configuration Register, Command Register, Packet Command Register, and Size Registers) for the Crypto, H1, and H2 cores by using the addresses (008, 0x09, and 0x0A), respectively. Need to execute the Wait instruction to make sure the cores are ready to accept data.
• Start new base	• Load Base Address Register into Source Address Register, load Size Register 0 into Source Size Register, and start transferring data into the CU InFIFO using the PLB Master Read interface until size reaches zero. • If OutDMA Enable bit of the Command Register is set, the PLB Master write interface of the control unit is also enabled at this time. • If the InDMA Enable bit of the command register is clear, no data transfer will occur. • Note that this instruction executes in one clock and no conditions are checked.

• Write_statblk	• Form command for the Status block based on the information specified in the Packet Command Register. • Wait until the Status block is not busy and then write the command to the Status Command Register. • Writing to the Status Command Register also causes the Status block to load the buffered status destination address to the current destination address register. • This instruction also generates pulses for H1 and H2, if the cores are enabled, to latch in the expected ICV from the respective buffers.
Write_data [write cfg][offset reg]	• [write cfg] - 6 bits wide (active high) • Bit 5 H1 1CV Write Enable -- enable writing to H1 Expected ICV Register. • Bit 4 H1 InFIFO Write Enable -- enable writing to H1 InFIFO. • Bit 3 H2 ICV Write Enable -- enable writing to H2 Expected ICV-Register. • Bit 2 - H2 InFIFO Write Enable -- enable writing to H1 InFIFO. • Bit I Crypto InFIFO Write Enable -- enable writing to Crypto InFIFO. • Bit 0 CU 0utFIFO Write Enable - enable writing to CU OutFIFO. • [offset reg] - 3 bits wide (selects one of 8 offset registers) • Broadcast the number of words specified by the selected offset register from the CU InFIFO to the destinations specified by the write configurations. • If the destination is a FIFO, the FIFO full flag is checked before writing. • Byte masking for H1 and H2 takes effect while executing this instruction, if enabled.
Wait [condition]	• [condition] - 8 bits wide • bit 7 - H1 not busy • bit 6 - H1 buffer not full

	• bit 5 - H1 InFIFO available • bit 4 - H2 not busy • bit 3 - H2 buffer not full • bit 2 - H2 InFIFO available • bit I - Crypto not busy • bit 0 - CU OutFIFO empty • Wait until the specified conditions are met.
Write [function select]	• [function select] - 3 bits wide • Bit 0 - Write Status Destination Address -- compute and write the destination address for the Status block. • Bit I - Write All -- command the control unit to write the remaining content of the OutFIFO to memory. If sequential output is enabled, this command is necessary (even if OutFIFO has nothing to write out) in order to pass on the token. • Bit 2 - Reset OutFIFO – generates pulse to reset CU OutFIFO. • Note that these functions execute in one clock and no conditions are checked.
Stop	• Reset CU InFIFO. • Reset Mask RAM read address pointer and mask logic. • Wait for CU OutFIFO to finish writing out data. • Start processing the next command if applicable (depend on control bits in the Command Register and Configuration Register)

Command FIFO Format

The Command FIFO format is 8 bits deep and 41-bit wide. The following is the format:

• Bits [40:32] specify the instruction RAM offset. This field is updated by writing to PLB address 0x040 with the offset on bits [8:0] of the data bus. This field is stored in a 9-bit register and is entered into the Command FIFO when the packet control structure address is updated.

- Bits [31:01] specify the packet control structure address. Writing to PLB address 0x041 causes the entire 41 bits to be loaded into the Command FIFO.

Outbound Packet Processing Instructions

The following Table C provides a representative example of instructions used in outbound packet processing:

Table C

Instructions	Packet Control Structure
load [39]	
store [base addr]	base addr
storex [offset reg 2, 41	offset 2 [begin of Hl]
	offset 3[begin of Encr]
	offset 4[end of Encr & Hl, begin of H1 result]
	offset 5[begin of H2 result]
storex [size reg 0, 4]	size 0[original packet data]
	size 1 [112]
	size 2[H1]
	size 3[Encr]
storex [mask FIFO, 21	mask bytes[O (leftmost) - 3 11
	mask bytes[32 – 631
storex [crypto key reg 0, 61	3DES key 1 [63:32]
	3DES key 1[31:001
	3DES key 2 [63:32]
	3DES key 2[31:001
	3DES key 3 [63:321
	3DES key 3[31:001
storex [crypto iv reg 0, 21	3DES IV [63:32]
	3DES IV P 1:001

storex [H1 hash reg 0, 51	HMAC I Inner IV, word 0
	HMAC I Inner IV, word 1
	HMAC I Inner IV, word 2
	HMAC I Inner IV, word 3
	HMAC I Inner IV, word 4
storex [Hl IVouter reg 0, 5]	HMAC I Outer IV, word 0
	HMAC I Outer IV, word 1
	HMAC I Outer IV, word 2
	HMAC I Outer IV, word 3
	HMAC I Outer IV, word 4
storex [H2 hash reg 0, 5]	HMAC 2 Inner IV, word 0
	HMAC 2 Inner IV, word I
	HMAC 2 Inner IV, word 2
	HMAC 2 Inner IV, word 3
	HMAC 2 Inner IV, word 4
storex [H2 IVouter reg 0, 5]	HMAC 2 Outer IV, word 0
	HMAC 2 Outer IV, word I
	HMAC 2 Outer IV, word 2
	HMAC 2 Outer IV, word 3
	HMAC 2 Outer IV, word 4
	Comments
start new base	read pkt data into InFIFO
write_dest [crypto dest addr reg, r3]	(=destination addr + offset reg
write [crypto cmd reg, data]	start crypto(write cmd & size reg 3)
write_dest [H 1 des addr reg, r4]	(=destination addr + offset reg)
write [H1 cmd reg, data]	start H1 (write cmd & size reg 2)
write dest [H1 des addr reg, r5]	(=destination addr + offset reg)
write [H2 cmd reg, data]	start H2(write cmd & size reg 1)
write	start ICV check/status reporting (enable crypto, Hl

[ICVchecker/stat cmd reg, data]	& H2 status reporting - status will be written to [Control Structure address + a fixed offset])
write-until0 [offset reg 2, write cfg]	write CU InFIFO data to H2 and CU OutFIFO until reach offset 2 (H2 byte mask ena)
write-until0 [offset reg 3, write cfg]	write CU InFIFO data to H1, H2 and CU OutFIFO until reach offset 3 (H2 byte mask ena)
*wait [CU OutFIFO empty]	wait until CU OutFIFO is empty (for sequential outputs)
write_until0 [offset reg 4, write cfg]	write CU InFIFO data to crypto until reach offset 4; H1 & H2 receives data from crypto
wait [H2 not busy]	wait until H2 is done (for sequential outputs)
stop	

* If destination is RAM, remove these instructions to improve performance.

Inbound Packet Processing Instructions

The following Table D provides a representative example of instructions used in outbound packet processing:

Table D

Instructions	Packet Control Structure
load [39]	
store [base addr]	base addr
storex [offset reg 1, 51	offset I [begin of H2 ICV]
	offset 2[begin of H11
	offset 3[begin of Encr]
	offset 4[end of Encr & Ill, begin of Ill result]
	offset 5[end of H2]
storex [size reg 0, 3]	size O[entire packet data = H2 size]
	size 1[H1]
	size 2[Encr]
storex [mask FIFO, 2]	mask bytes[O (leftmost) - 3 1
	mask bytes[32 - 63]
storex [crypto key reg 0, 6]	3DES key 1 (63:32]
	3DES key 1[31:00]
	3DES key 2 [63:32]

	3DES key 2[31:00]
	3DES key 3 [63:321
	3DES key 3[31:00]
storex [crypto iv reg 0, 2]	3DES IV [63:321
	3DES IV [31:00]
storex [111 hash reg 0, 51	HMAC 1 Inner IV, word 0
	HMAC 1 Inner IV, word I
	HMAC I Inner IV, word 2
	HMAC 1 Inner IV, word 3
	HMAC I Inner IV, word 4
storex [111 outer reg 0, 5]	HMAC 1 Outer IV, word 0
	HMAC I Outer IV, word 1
	HMAC I Outer IV, word 2
	HMAC I Outer IV, word 3
	HMAC I Outer IV, word 4
storex [H2 hash reg 0, 5]	HMAC 2 Inner IV, word 0
	HMAC 2 Inner IV, word I
	HMAC 2 Inner IV, word 2
	HMAC 2 Inner IV, word 3
	HMAC 2 Inner IV, word 4
storex [H2 IVouter reg 0, 5]	HMAC 2 Outer IV, word 0
	HMAC 2 Outer IV, word 1
	HMAC 2 Outer IV, word 2
	HMAC 2 Outer IV, word 3
	HMAC 2 Outer IV, word 4
	Comments
start new base	read Pkt data into CU InFIFO
write-dest [crypto dest addr reg, r3]	(=destination addr + offset reg
write [crypto cmd reg, data]	start crypto(write cmd & size reg 2)
write [H1 cmd reg, data]	start H1(write cmd & size reg 1)
write [112 cmd reg,	start H2(write cmd & size reg 0)

data]	
write [ICVchecker/stat cmd reg, data]	start ICV check/status reporting (enable crypto, Hl, & H2 status reporting and H1 & H2 ICV checks - status will be written to [Control Structure address a fixed offset])
write_until0 [offset reg 1, write cfg]	write CU InFIFO data to H2 and CU OutFIFO until reach offset I (H2 byte mask ena)
write_until0 [offset reg 2, write cfg]	write CU InFIFO data to H2, H2 ICV Checker and CU OutFIFO until reach offset 2 (H2 byte mask ena)
write-until0 [offset reg 3, write cfg]	write CU InFIFO data to H 1, H2 and CU OutFIFO until reach offset 3 (H2 byte mask ena)
*wait [CU OutFIFO empty]	wait until CU OutFIFO is empty (for sequential outputs)
write-until0 [offset reg 4, write cfg] .	write CU InFIFO data to H 1, H2, and crypto until reach offset 4;
*wait [crypto not busy]	wait until crypto done writing out results(for seg. outputs)
write_until0[offset reg 5, write cfg]	write CU InFIFO data toH2, H1 ICV Checker and CU OutFIFO until reach offset 5;
*wait [CU OutFIFO empty]	wait until CU OutFIFO is empty (for sequential outputs)
stop	

- If destination is RAM, remove these instructions to improve performance.

<u>Control Unit Tables</u>

The following Tables E, F, and G provide representative examples of registers in the control unit.

<u>Table E</u>

Control Unit -- Configuration Register (Writing to this register is allowed only when the packet processor is not busy, except for the Terminate and Stop bits.)	
Bits	Description
31	Terminate. This bit set commands the packet processor to stop immediately. The PPC needs to set the terminate bits of the cores as well if the cores are to stop execution. This bit can be

	modified at any time.
30	Stop When Empty. This bit set commands the packet processor to stop (once it is done executing the current command) when the command FIFO becomes empty. This bit is used only when the Execute. until Stop bit of the command register is set and DMA is enabled. This bit can be modified at any time.
29	Stop When Done. This bit set commands the packet processor to stop processing commands from the command FIFO once it is done executing the current command. This bit is used only when the Execute until Stop bit of the command register is set and DMA is enabled. This bit can be modified at any time.
10	H2 InFIFO Early Release Enable. not available yet.
9	
8	HMAC OutDMA Enable. This bit set specifies to always write out H1 and H2 results (via the PLB master write interface). This bit clear specifies to write out H1 and H2 results only when it is an outbound packet and the OutDMA Enable bit of the CU Command Register is set.
7	Packet Processor Enable. This bit selects between the two write interfaces of the Crypto, Hl, and H2 cores. This bit set selects to use the interface to the packet processor control unit for write operations to the cores. This bit clear selects to use the PLB slave interface of the cores.
6	Instruction RAM Configuration Enable. This bit set allows the PPC to write to the Instruction RAM. This bit clear inhibits writing to the Instruction RAM.
5	Reset Command FIFO. This bit set commands the packet processor to reset the Command FIFO. This bit should be clear during normal operation
4	Sequential Output Disable. This bit clear commands the packet processor to write out packet processing results sequentially. This bit set commands the packet processor to write out results as they become available (not necessarily sequential).
3:2	Status Local Bus Priority[1:0]. These bits specify the priority of the status block on the local bus. 00 specify lowest priority; 11 specify highest priority.
1:0	FIFO Local Bus Priority[1:0]. These bits specify the priority

	of the PLB Master interfaces used for data transfer to/from the CU InFIFO and OutFIFO on the local bus. 00 specify lowest priority; 11 specify highest priority.

Table F

Control Unit - Command/Status Register (Writing to this register initiates packet processing. Writing to this register is not allowed when the busy bit is set.)	
Register Bits	Description
31	Busy. This bit set indicates that the packet processor busy. This bit clear indicates that the packet processor is ready . This bit is read only.
30	Command FIFO Full. Ibis bit set indicates the command FIFO is full. This bit is read only.
29	Command FIFO Empty. This bit set indicates the command FIFO is empty. This bit is read only.
5:4	Execution Mode[1:0]. 00 -- Execute Until Stop. This bit set specifies to continuously execute commands from the command FIFO until one of the Stop bits in the configuration register is set. 01 -- Execute Until Empty. This bit set specifies to continuously execute commands from the command FIFO until the FIFO becomes empty. Ix -- Execute One Command. This bit set specifies to execute one command from the command FIFO and then stop.
3	OutDMA AutoIncrement Disable. This bit clear specifies to increment the destination address when using the PLB master, write interface to transfer data from the control unit OutFIFO, status block, and the cores. This bit set specifies not to increment the destination address.
2	OutDMA Enable. This bit set enables the control unit OutFIFO, status block, and the Crypto core to write out the results via their PLB master write interfaces. This bit set also enables H1 and H2 to write out results for outbound packet. This bit clear disables all PLB master write interfaces.
1	InDMA AutoIncrement Disable. This bit clear specifies to increment the source address when using the PLB master read

	interface to transfer data into the control unit, Crypto, H 1, and H2 InFIFOs. This bit set specifies not to increment the source address.
0	InDMA Enable. This bit set enables the control unit to execute commands from the command FIFO and to use the PLB master read interface to transfer data from memory to the control unit InFIFO. If this bit is clear, the PPC writes the instruction RAM offset, the packet control structure, and packet data to the control unit.

Table G

Control Unit - Packet Command Register (This register is written by the control unit only.)	
Register Bits	Description
31	H1 Mask Enable - This bit set enables byte masking of up to 256 bytes of data written to H1 InFIFO, beginning from the base address. Byte masking occurs while executing the Write-data instruction. This bit clear specifies not to mask data.
30:28	H1 Mask Size[2:0] - These bits selects the number of mask words to be applied. "000" selects to enable byte masking of the first 32 bytes of packet data. "001" selects to enable byte masking of the first 64 bytes of packet data. ..."111" selects to enable byte masking of the first 256 bytes of packet data.
27	H1 Initialize Hash. This bit set specifies to use the default initial value specified by the technique as the starting hash value. Ibis bit clear specifies to use the value currently in the Hash Registers as the starting hash value.
26	H1 Final Block. This bit set specifies to append padding and complete the hash operation. If the HMAC technique is selected, the core will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding/length should be appended. Note that size must be multiples of 512 bits if this bit is not set.
25:24	H1 Technique [1:0]. 00 -- MD5 01 -- SHA-1 10 -- HMAC-MD5

	11 -- HMAC-SHA-1
23	H2 Mask Enable - This bit set enables byte masking of up to 256 bytes of data written to H2 InFIFO, beginning from the base address. Byte masking occurs while executing the Write-data instruction. This bit clear specifies not to mask data.
22:20	H2 Mask Size[2:0] - These bits selects the number of mask words to be applied. "000" selects to enable byte masking of the first 32 bytes of packet data. "001" selects to enable byte masking of the first 64 bytes of packet data. selects to enable byte masking of the first 256 bytes of packet data.
19	H2 Initialize Hash. This bit set specifies to use the default initial value specified by the technique as the starting hash value. This bit clear specifies to use the value currently in the Hash Registers as the starting hash value.
18	H2 Final Block. This bit set specifies to append padding and complete hash. If HMAC technique is selected, the core will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding/length should be appended. Note that size must be multiples of 512 bits this bit is not set.
17:16	H2 Technique[1:0]. 00 -- MD5 01 -- SHA-1 10 -- HMAC-MD5 11 -- HMAC-SHA-I
15	H1 Length/Ipad/Opad Select. When the initialize bit is set and the final bit is clear, this bit is used to select ipad/opad: this bit set specifies to use the HMAC ipad to perform the HMAC initialization only command; this bit clear specifies to use the HMAC opad to perform the HMAC initialization only command Otherwise, this bit is used to select the source of message length: this bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use Size as the length of the message.
14	H2 Length/Ipad/Opad Select. When the initialize bit is set and the final bit is clear, this bit is used to select ipad/opad: this bit set specifies to use the HMAC ipad to perform the HMAC initialization only command; this bit clear specifies to use the HMAC opad to perform the HMAC initialization only command. Otherwise, this bit is used to select the source of

	message length: this bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use Size as the length of the message.
13	Crypto 3DES Keys for Decryption. This bit set specifies that keys 1,2, and 3 are in the order for decryption and that the order of the keys should be reversed if encryption mode is selected. This bit clear specifies that keys 1,2, and 3 are in the order for encryption and that the order of the keys should be reversed if decryption mode is selected.
12	Crypto Initialize RC4. This bit set specifies to initialize the RC4 Sbox with the key loaded in the key registers. This bit clear specifies to use the key stream which is currently in the RC4 Sbox Registers.
11:8	Crypto Technique[3:0]. bit 3: 1 = RC4 0--DES/3DES; bit 2: 1= 3DES O= DES; bit 1: 1 = ECB 0= CBQ bit 0: 1= decryption 0= encryption
7	Core InDMA Enable. This bit set enables the Crypto, Hl, and H2 cores to fetch their own data (for the InFIFOs) using their PLB Master read interfaces. The daisy chain bus is disabled and all cores operate independent from each other. The status of the cores may still be reported by the status block. This bit clear specifies that the cores receive their data from the control unit.
6	Crypto Enable. This bit set indicates that the Crypto core is required for the processing of this packet. This bit clear indicates that the Crypto core is not required for the processing of this packet. This bit is used to enable the status reporting for Crypto, determine the destination address for status block, and gate write operations to Crypto registers and InFIFO.
5	H1 Enable. This bit set indicates that H1 core is required for the processing of this packet. This bit clear indicates that H1 core is not required for the processing of this packet. This bit is used to enable the status reporting for Hl, determine the destination address for status block, gate write operations to H1 registers and InFIFO, and enable daisy chain outputs.
4	H2 Enable. This bit set indicates that H1 core is required for the processing of this packet. This bit clear indicates that H1 core is not required for the processing of this packet. This bit is used to enable the status reporting for H1, determine the

	destination address for status block, gate write operations to H1 registers and InFIFO, and enable daisy chain outputs.
3:2	Mode Select[1:0] -00 - selects to use only the leftmost 96 bits of hash results. 01 - selects to use only the leftmost 128 bits of hash results. Ix - selects to use the entire hash results.
I	Inbound Packet. This bit set indicates that this is an inbound packet. This bit clear indicates that this is an outbound packet.
0	Endian. This bit set specifies that the packet data to be processed is in little endian format. This bit clear specifies that the packet data to be processed is in big endian format. This bit also specifies the endian format for outputs from the packet processor, except for authentication values and status. Note that data in the Mask RAM must before packet data in big endian format.

Status Block

The following Table H provides a representative example of the status block command and status register:

Table H

Status Block Command/Status Register	
Register Bits	Description
31	Completed packet processing. This bit set indicates that packet processing has completed. This bit clear indicates that the packet is not yet processed or that processing has not completed. This bit is cleared when writing to this register.
30:24	Reserved. Read as zero.
23	Crypto Status Enable. This bit set specifies to report Crypto status.
22	Crypto Done. This bit set indicates that the Crypto block has completed processing. This bit is cleared when writing to this register.

21	Crypto Key error. This bit indicates either RC4 key length error or DES/3DES key parity error, depending on the technique, as follows: RC4: This bit set indicates that the key length is 0 or greater than 32 bytes when the Initialize RC4 bit is set. DES/3DES: This bit set indicates that the keys (key 1 for DES; key 1/2/3 for 3DES) contain parity error. This bit is cleared when each byte of the keys has odd parity. This error does not stop processing. This bit is cleared when writing to this register.
20:16	Reserved. Read as zero.
15	H2 Status Enable. This bit set specifies to report H2 status.
14	H2 Done. This bit set indicates that the H2 has completed processing. This bit is cleared when writing to this register.
13	H2 Size Error. This bit set specifies that Size is not an even multiple of 64 bytes and the Final Block bit is low; no hashing will be performed. This bit clearly specifies that Size is valid. This bit is cleared when writing to this register
12	H2 ICV Check Enable. This bit set specifies to compare H2 result with H2 authentication value from the inbound packet.
11	H2 ICV Check Status. This bit set indicates that H2 ICV match. This bit clear indicates the comparison failed. This bit is cleared when writing to this register.
10:8	Reserved. Read as zero.
7	H1 Status Enable. This bit set specifies to report Hl status. If this bit is clear, ignore the H1 status bits..
6	Hl Done. This bit set indicates that the H1 has completed processing. Ibis bit is cleared when writing to this register.
5	H1 Size Error. This bit set specifies that Size is not an even multiple of 64 bytes and the Final Block bit is low; no hashing will be performed. This bit clearly specifies that Size is valid. This bit is cleared when writing to this register
4	H1 ICV Check Enable. This bit set specifies to compare H1 results with H1 authentication value from the inbound packet.
3	H1 ICV Check Status. This bit set indicates that H1 ICV match. This bit clear indicates the comparison failed. This bit is cleared when writing to this register
2: 0	Reserved. Read as zero.

IP Security Encryption Techniques

The goal of the IP security encryption techniques is to do the following transformations:

- DES - ECB and CBC
- 3DES - ECB and outer CBC
- RC4

The code performance of the IP security encryption techniques are as follows:

• DES:	8 bytes/ 8 clocks
• 3DES:	8 bytes/ 24 clocks
• RC4:	I byte / I clock (Initialization: 384 clocks)

The following Table I provides a representative example of additional information about the IP security encryption techniques:

Table I

Command	Technique	RC4 Init	3DES keys for decr	Data req'd	Function
DES CBC complete	DES CBC encr/decr	x	x	key, IV, text	Perform complete DES CBC encryption or decryption
DES ECB complete	DES ECB encr/decr	x	x	key, text	Perform complete DES ECB encryption or decryption.
3DES outer	3DES CBC	x	y/n	key,	Perform complete

CBC complete	encr/decr			IV, text	3DES outer CBC encryption or decryption.
3DES ECB complete	3DES ECB encr/decr	x	y/n	key, text	Perform complete 3DES ECB encryption or decryption.
RC4 complete	RC4 encr/decr	y	x	key length, key, text	Perform complete RC4 encryption or decryption.
RC4 without init	RC4 encr/decr	n	x	text	Perform RC4 encryption or decryption without initializing (continue from preceding processing).

In one preferred representative example, the Crypto block 108 performs RC4, DES and 3DES encryption techniques. Input text is either read in by the PLB Master read interface or provided by the packet processor control unit 106. The Crypto block begins execution following the writing to the Command/Status Register. The completion of processing is indicated by the transition of the Busy bit in the Command/Status Register from a one to a zero.

As a representative example, the PLB Address for the Crypto Configuration Register is: 0x000200, and this register can be updated at any time. The following Table J provides a representative example of information about the crypto configuration register:

Table J

Register Bits	Description
26:25	Local Bus Priority[1:0]. These bits specify the priority of this block on the local bus. 00 specify lowest priority;

	11 specify highest priority.
24	Terminate. This bit set specifies to terminate the current process. This bit should normally be set to low. Writing to this bit is always allowed.

Further to the above representative example, the PLB Address of the Key Length Register is: 0x000204. This 6-bit register contains the key length in number of bytes. Maximum key length is 32 bytes. READ and WRITE are not allowed while RC4 Init Busy is high. The PLB Addresses of the Key Registers are: 0000208, 0x00020C, 0000210, 0000214, 0000218, 000021C, 0000220, 0x000224. These 8 32-bit registers store the key. The register at PLB address 0x208 holds the most significant word, with the leftmost character of the key in the 31:24 bit position.

For DES and 3DES, the following are keys and associated registers:

- Key 1 [63:0] is stored in registers at PLB address [0000208 000020C].
- Key 2 [63:0] is stored in registers at PLB address [0000210 0000214].
- Key 3 [63:0] is stored in registers at PLB address (0000218 000021C].

Each 64-bit key is loaded into the DES/3DES engine the same clock cycle when a write to the later address (000020C, 0000214, 0x00021C) is performed. Moreover, READ and WRITE operations are not allowed while RC4 Init Busy is high.

In the above-representative example, the PLB Addresses for the DES IV Registers are: 0000228, 0x0022C. These 2 32-bit registers store the initialization vector for DES/3DES CBC mode. The register at PLB address 0x00228 holds the most significant 32-bit word, with the leftmost character of the vector in the 31:24 bit position. WRITE operations are not allowed while Busy is high.

Further to the above representative example, the PLB Address of the Source Address Register is: 0x000230. This 32-bit source address register points to the location where the input data to the crypto block will be read. The PLB Address of the Destination Address Register is:

0x000234. This 32-bit destination address register points to the location where the output data from the hash block will be written.

Further to the above example, the PLB Address for the Crypto Command/Status Register is : 0x000238. Writing to this register starts the Crypto block processing. Writing to this register is inhibited while the busy bit is set. A representative example of this register is defined in Table K, which the "crypto" command/status register information.

Table K

Register Bits	Description
31	Busy. This read-only bit set indicates that the Crypto block is processing. This bit clear indicates that the Crypto block is idle and is ready for a new command.
30	RC4 Initialization Busy. This status bit is set while RC4 is initializing.
29	Key error. This bit indicates either RC4 key length error or DES/3DES key parity error, depending on the technique, as follows: RC4: This bit set indicates that the key length is 0 or greater than 32 bytes when the Initialize RC4 bit is set. Neither initialization nor encryption will be performed. Writing to this register clears this bit. DES/3DES: This bit set indicates that the keys (key I for DES; key 1/2/3 for 3DES) contain parity error. This bit is cleared when each byte of the keys (key 1 for DES; keys 1&2&3 for 3DES) is of odd parity. This error does not stop processing.
28	DCout To H1 Enable. This bit set enables the core to pass the resulting text to H1 on the daisy chain bus.
27	DCout To H2 Enable. This bit set enables the core to pass the resulting text to H2 on the daisy chain bus.
26	OutDMA AutoIncrement. This bit set specifies to enable the PLB master write interface to increment destination address.
25	OutDMA enable. This bit set specifies to use the DMA bus for output transfers.
24	3DES Keys for Decryption. This bit set specifies that keys 1,2,

	and 3 are in the order for decryption and that the are in the order for encryption and that the order of the keys should be reversed if technique[0] is set for decryption.
23:20	Technique[3:0] bit 3: 1= RC4 0--DES/3DES; bit 2: I= 3DES 0= DES; bit 1: I= ECB 0-- CBQ bit 0: 1= decryption 0-- encryption
19	Initialize RC4. This bit set specifies to initialize the RC4 Sbox with the key loaded in the key registers. This bit clear specifies to use the key stream which is currently in the RC4 Sbox Registers.
18	InDMA AutoIncrement. This bit set specifies to enable the PLB master read interface to increment destination address.
17	InDMA enable. This bit set specifies to use the DMA bus for input transfers.
16	Endian. This bit set specifies that the format of the data to/from the 64-bit FIFO is little endian. 'Ibis bit clear specifies that the format of the data to/from the 64-bit FIFO is big endian. Note that endianness refers to the byte ordering within 32-bit words.
15:0	Size. Writing to these bits specifies the number of bytes to be encrypted. Reading of this field indicates the number of bytes remaining to be processed and read out. For DES/3DES, size must be an integral of 64-bit words.

Further to the above example, the PLB Address for the FIFO is: 0x100000 - 0x1FFFFF. The InFIFO and OutFIFO may be accessed using an address in the above address range. The Endian bit in the Command/Status register must be written to indicate the byte ordering of the data to be written to and read from the FIFOs. Data is written into the InFIFO. Data is read from the OutFIFO. The FIFOs are accessible via the PLB bus only when the DMA enable bit is set low.

IP Security HMAC Block

The IP security HMAC blocks may use one or a combination of the following authentication techniques:

- HMAC-MD5 (-96)

- HMAC-SHA-I (-96)
- MD5
- SHA-1

The HMAC technique is a keyed hashing technique for message authentication. The HMAC technique operates in combination with a hash function and requires a key. The following equation describes the HMAC technique:

Hash (Key XOR opad, Hash (Key XOR ipad, text));

For higher performance, the technique may generate the inner IV (Hash (Key XOR ipad)) and outer IV (Hash (Key XOR opad)) for each key and store them for later use. When keyed hashing is required, the stored inner IV and outer IV stored for that key are provided, and the HMAC update or HMAC update and final command are used to perform the hash. Table L provides a representative example of authentication commands.

Table L

Cmd	Tech.	Init	Final Blk	Opad / Ipad	Len Src	Data req'd	Function
HMAC complete	HMAC MD5/ SHA-1	y	y	x	size/ len	key, text	Perform complete HMAC func. (only support key size of 160 bits for SHA- 1 and 128 bits for MD5)
HMAC inner init	HMAC MD5/	y	n	ipad	x	key	w/default IV, Key XOR

	SHA-1						ipad, hash 64B.
HMAC outer init	HMAC MD5/ SHA-1	y	n	opad	x	key	Generate IVouter by init hash w/default IV, Key XOR opad, hash 64B
HMAC update	HMAC MD5/ SHA-1	n	n	x	x	IVinne, text	Hash text using provided IVinner
HMAC update and final	HMAC MD5/ SHA-1	n	y	x	size/ len	Ivinne, Ivouter, text	Hash text, with padding appended, using the provided IVinner, and then perform the outer hash from IVouter
Hash complete	MD5/ SHA-1	y	y	X	size/ len	text	Perform complete hash function
Hash init and update	MD5/ SHA-1	y	n	X	x	text	Initialize with hash default IV an then hash text
Hash update and final	MD5/ SHA-1	n	y	X	size/ len	IV, text	Hash text, with padding appended, using the provided IV.

The HMAC block performs Keyed-hashing for message authentication and hash techniques SHA-1 and MD5. Input data is either read in by the PLB Master read interface or provided by the packet processor control unit. The HMAC block begins execution following the writing to the Command/Status Register. Processing is completed when the Busy bit in the Command/Status Register transitions from a one to a zero.

Further to the above example, the PLB Address of the HMAC Configuration Register is [5:2]: 0x0. This register can be updated at any time. The following Table M provides a representative example of HMAC configuration register information.

Table M

Register Bits	Description
26:25	Local Bus Priority[1:0]. These bits specify the priority of this block on the local bus. 00 specify lowest priority; 11 specify highest priority.
24	Terminate. This bit set specifies to terminate the current process. This bit should normally be set to low. Writing to this bit is always allowed.

The PLB Addresses for the Length Registers are- 0x1, 0x2. The 61-bit length indicates the number of bytes in the message. Register at PLB address 0x1 contains the lower 32-bit word. Register at PLB address 0x2 contains the upper 29 bits.

The PLB Addresses for the Outer IV Register are: 0x3 - 0x7. These registers store the outer initial value for HMAC.

The PLB Addresses for the Hash Register are: 0x8 - 0xC. These registers store the hash value. Reading these registers returns the hash value of the last 512 bit block, which has been hashed. Writing to these registers sets the initial hash value for the next block to be hashed (this is also referred to as the inner initial value for HMAC). The register at PLB address 0x8 contains the most significant word. The register at PLB address 0xC is not used for MD5. Hash values read from the registers should be written back to the registers the same way. The final result begins with the high-order byte of regAA (PLB address 0x8) and ends with the low-order byte of regDD (PLB address 0xB) for MD5 and of regEE (PLB address 0xC) for SHA-1, with the following:

```
MD5=>       A3 A2 Al... B3 ... C3 ... D3 D2 DI D0
SHA-1->     A3 A2 Al ..B3  ..C3 ...D3  .............E3 E2 El E0
```

(i.e., byte swapping is automatically done for the final MD5 block to achieve the above result).

The PLB Address for the Source Address Register is: 0xD. This 32-bit source address register points to the location where the input data to the hash block will be read.

The PLB Address for the Destination Address Register is: 0xE. This 32-bit destination address register points to the location where the output data from the hash block will be written.

The PLB Address for the HMAC Command/Status Register is: 0xF. Writing to this register starts the HMAC block processing. Writing to this register is inhibited while the busy bit is set. The following Table N contains a representative example of HMAC command/status register information.

Table N

Register Bits	Description
31	Busy. This read-only bit set indicates that the core is processing. This bit clear indicates that the core is idle and is ready for a new command.
30	Size Error. This bit set specifies that Size is not an even multiple of 64 bytes and the Final Block bit is low; no hashing will be performed. This bit clear specifies that Size is valid. This bit is set to high by the hash block. Writing to this register clears this bit.
29:28	Mode Select[1:0]. 00 use the leftmost 96 bits of the hash value 01 use the leftmost 128 bits of the hash value Ix use the entire hash value (160 bits for SHA-1, 128 bits for MD5)

27	DCout Enable. This bit set enables the core to pass the hash value to the next core on the daisy chain bus. This bit clear specifies not to write out the hash value. The number of words transferred is selected by the mode select bits. This bit is for H1 only. This bit should be clear for H2.
26	OutDMA AutoIncrement. This bit set enables the PLB master write interface to increment the destination address.
25	OutDMA enable. This bit set specifies to use the DMA bus for output transfers.
24	InFIFO Early Release Enable. This bit set specifies to release the InFIFO when it is only waiting for data from H1 (as in [AH] [ESP with authentication]). This occurs when the remaining size to be processed equal to the number of words specified by mode select bits. This bit is for H2 only. This bit should be clear for H1.
23	Initialize Hash. This bit set specifies to use the default initial value specified by the technique as the starting hash value. This bit clear specifies to use the value currently in the Hash Registers as the swing hash value.
22	Final Block. This bit set specifies to append padding and complete hash. If HMAC technique is selected, the core will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding/length should be appended. Note that size must be multiples of 512 bits this bit is not set.
21:20	Technique[1:0]. 00 -- MD5 01 -- SHA-I 10 – HMAC-MD5 11 – HMAC-SHA-1
19	Length/Ipad/Opad Select. When the initialize bit is set and the update and finish bits are clear, this bit is used to select ipad/opad: this bit set specifies to use the HMAC ipad to perform the HMAC initialization only command; this bit clear specifies to use the HMAC opad to perform the HMAC initialization only command. Otherwise, this bit is used to select the source of message length: this bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use Size as the length of the message.

18	InDMA AutoIncrement. This bit set enables the PLB master read interface to increment the source address.
17	InDMA enable. This bit set enables the core to DMA in the input text. This bit clear specifies that input data will come from either the control unit or the daisy chain bus (when the direct path signal is active).
16	Endian. This bit set specifies that data to be transferred into the InFIFO is in little endian format. This bit clear specifies that data to be transferred into the InFIFO is in big endian format. Endian conversion is done here.
15:0	Hash Size. These bits specify the number of bytes to be hashed. Writing to these bits specifies the number of bytes to be hashed. Reading from this register indicates the number of bytes remaining to be hashed. When performing HMAC initialization only command (to generate HMAC IVinner/IVouter), this size indicates the number of bytes in the key.

Conclusion

This concludes the description of the preferred embodiment of the invention. The following describes some alternative embodiments for accomplishing the present invention. For example, any type of computer, such as a mainframe, minicomputer, or personal computer, or computer configuration, such as a timesharing mainframe, local area network, or standalone personal computer, could be used with the present invention.

The foregoing description of the preferred embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.

WHAT IS CLAIMED IS:

1. A packet processor comprising:
a control unit having a data input path;
at least one encryption unit;
a first authentication unit;
a second authentication unit;
a local data path, independent of the data input path to the control unit, coupling the control unit to each of the encryption and authentication units; and
a second data path from the encryption unit to each authentication unit, including a data path from the first authentication unit to the second authentication unit.

2. A packet processor as recited in claim 1, wherein said data input path of the control unit is coupled to a processor bus and each of said encryption and authentication units comprises a data input path coupled to the processor bus.

3. A packet processor as recited in claim 1, wherein said data input path of the control unit is coupled to a processor bus and each of said encryption and authentication units comprises a data input path to the processor bus and means for reading and writing data on the processor bus.

4. A packet processor as recited in claim 1, wherein said second data path comprises a daisy-chain connection between the encryption and authentication units.

5. A method of processing data packets comprising:

coupling a control unit to a first data path;

receiving first and second data packets in the control unit from the first data path;

providing a plurality of processing units in data communication with the control unit over a second data path, independent of the first data path, said processing units including at least one encryption unit and at least one authentication unit;

providing data of the first data packet from the control unit to one of the processing units, over the second data path;

processing said data from the first data packet with said one of the processing units to provide output data for the first data packet from said one of the processing units;

communicating said output data for the first data packet from said one of the processing units to another of the processing units for further processing; and

providing data from the second data packet to said one of the processing units, while said other processing unit further processes the output data for the first data packet.

6. A method as recited in claim 5, wherein said one of the processing units comprises an encryption unit and said other of said processing units comprises an authentication unit.

7. A method as recited in claim 5, wherein said at least one authentication unit comprises a first and second authentication units.

8. A method as recited in claim 5, wherein said step of communicating the output data from one of the processing units to another of the processing units comprises communicating said output data over a daisy-chain connection between said processing units.

9. A method of processing data in a computer, the method comprising the steps of:
performing encryption on a first data packet; and
after completion of the encryption of the first data packet,
performing authentication of the first data packet, and
performing encryption of a second data packet prior to completion of authentication of the first data packet.

10. The method of claim 9, wherein the authentication is a first authentication, further comprising the step of performing a second authentication on the first data packet of data.

11. The method of claim 10, wherein the first authentication is performed on the encrypted first data packet.

12. The method of claim 10, wherein the first authentication appends data to the encrypted first data packet.

13. The method of claim 12, wherein the second authentication is performed on the encrypted first data packet and the appended data.

14. The method of claim 10, further comprising the step of performing the encryption of the second data packet after beginning the second authentication of the first data packet.

15. A method of processing data in a computer, the method comprising the steps of:

encrypting a first data packet with an encryption module;

authenticating the encrypted first data packet with a first authentication module;

encrypting a second data packet with the encryption module while authenticating the first data packet with the first authentication module; and

authenticating the second data packet with the first authentication module.

16. An apparatus for processing data, comprising:

a computer having a data storage device connected thereto, wherein the data storage device stores a data;

one or more computer programs, performed by the computer, for performing encryption on a first data packet, and, after completion of the encryption of the first data packet, performing authentication of the first data packet, and performing encryption of a second data packet prior to completion of authentication of the first data packet.

17. The apparatus of claim 16, wherein the authentication is a first authentication, further comprising means for performing a second authentication on the first data packet of data.

18. The apparatus of claim 17, wherein the first authentication is performed on the encrypted first data packet.

19. The apparatus of claim 17, wherein the first authentication appends data to the encrypted first data packet.

20. The apparatus of claim 19, wherein the second authentication is performed on the encrypted first data packet and the appended data.

21. The apparatus of claim 17, further comprising the means for performing the encryption of the second data packet after beginning the second authentication of the first data packet.

22. An apparatus for processing data, comprising:

a computer having a data storage device connected thereto, wherein the data storage device stores a data;

one or more computer programs, performed by the computer, for encrypting a first data packet with an encryption module, authenticating the encrypted first data packet with a first authentication module, encrypting a second data packet with the encryption module while authenticating the first data packet with the first authentication module, and authenticating the second data packet with the first authentication module.

23. An article of manufacture comprising a computer program carrier readable by a computer and embodying one or more instructions executable by the computer to perform method steps for processing data, the method comprising the steps of:

performing encryption on a first data packet; and

after completion of the encryption of the first data packet,

performing authentication of the first data packet, and

performing encryption of a second data packet prior to completion of authentication of the first data packet.

24. The article of manufacture of claim 23, wherein the authentication is a first authentication, further comprising the step of performing a second authentication on the first data packet of data.

25. The article of manufacture of claim 24, wherein the first authentication is performed on the encrypted first data packet.

26. The article of manufacture of claim 24, wherein the first authentication appends data to the encrypted first data packet.

27. The article of manufacture of claim 26, wherein the second authentication is performed on the encrypted first data packet and the appended data.

28. The article of manufacture of claim 24, further comprising the step of performing the encryption of the second data packet after beginning the second authentication of the first data packet.

29. An article of manufacture comprising a computer program carrier readable by a computer and embodying one or more instructions executable by the computer to perform method steps for processing data, the method comprising the steps of:

encrypting a first data packet with an encryption module;

authenticating the encrypted first data packet with a first authentication module;

encrypting a second data packet with the encryption module while authenticating the first data packet with the first authentication module; and

authenticating the second data packet with the first authentication module.

30. A method of processing data packets comprising:

coupling a control unit to a first data path;

receiving a first dat packet in the control unit from the first data path;

providing a plurality of processing units in data communication with the control unit over a second data path, independent of the first data path, said processing units including at least one encryption unit and at least one authentication unit;

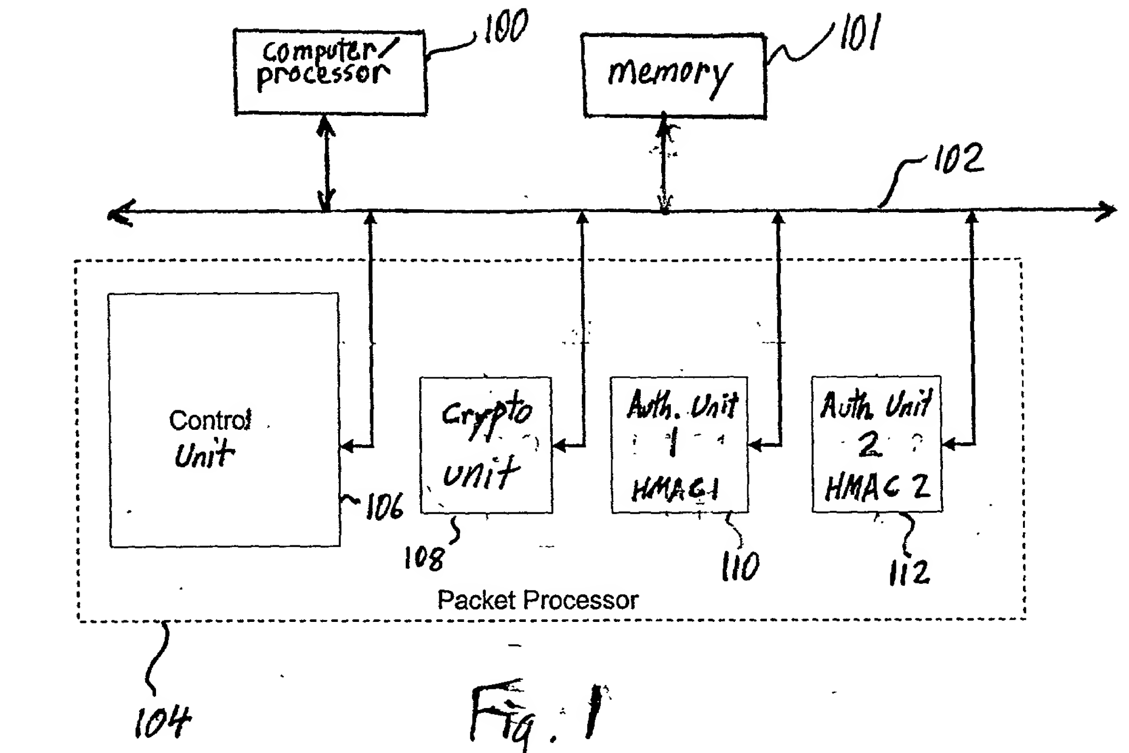
providing data of the first data packet from the control unit to multiple processing units, over the second data path;

processing said data from the first data packet with said multiple processing units in parallel.

31. A method as recited in claim 30, wherein said plurality of processing units comprises at least one encryption unit and a plurality of authentication units.

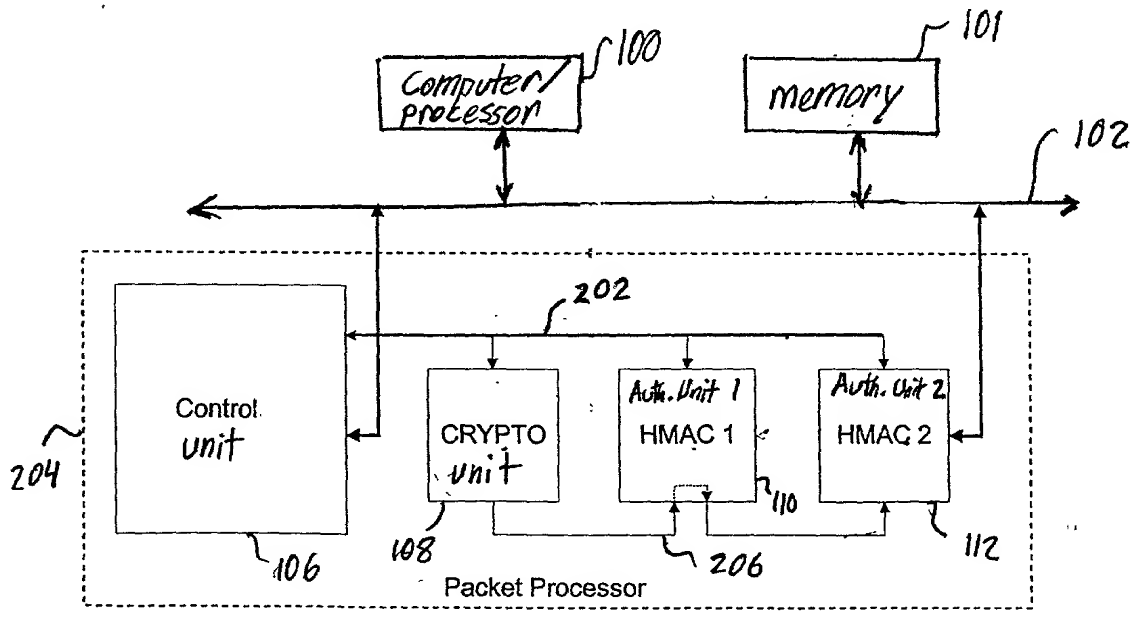
ABSTRACT OF THE DISCLOSURE

A method, apparatus, and article of manufacture for a computer implemented packet processor. The packet processor processes packets in parallel. In particular, the packet processor performs a combination of encryption and authentication on data packets. The encryption and authentication processing of a second data packet may begin before the encryption and authorization processes of a first data packet have completed.



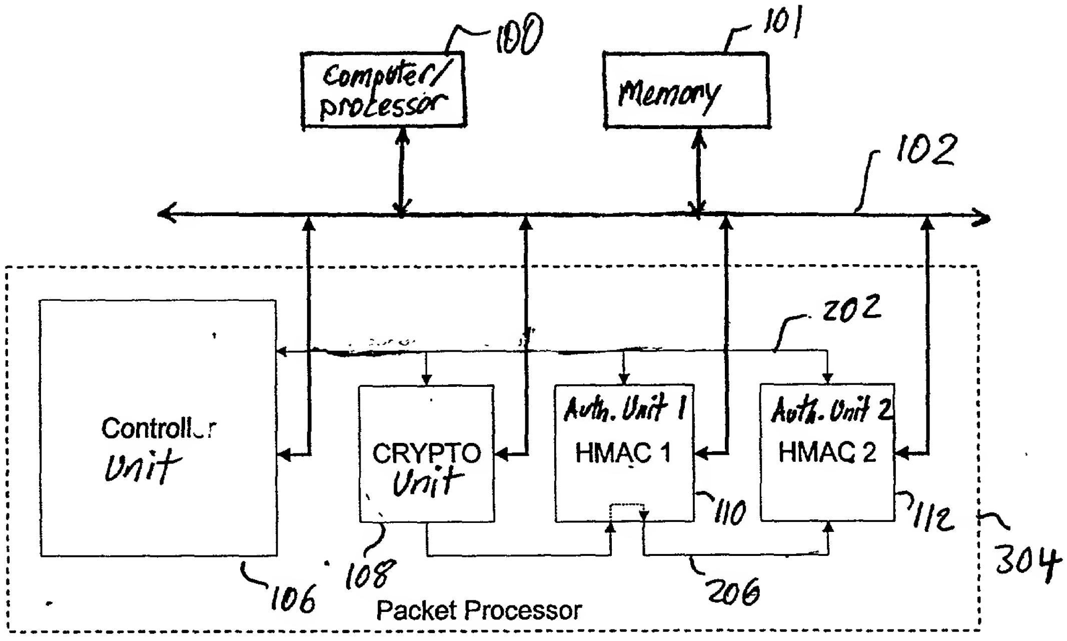
computer/ processor
100
memory
101
102
Control Unit
106
Crypto unit
108
Auth. Unit 1 HMAC 1
110
Auth. Unit 2 HMAC 2
112
Packet Processor
104

Fig. 1



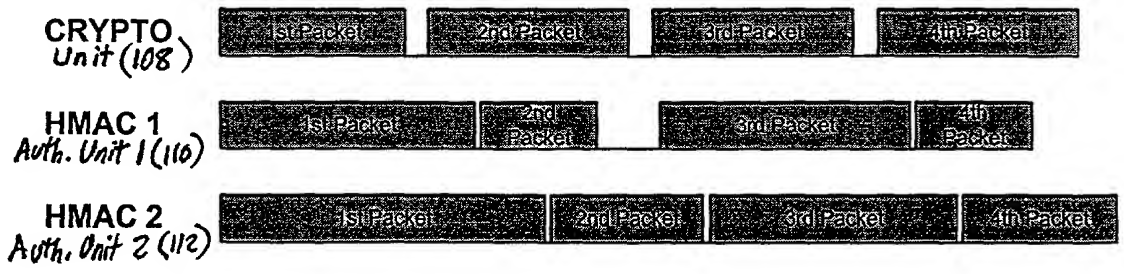
Computer/ processor
100
memory
101
102
202
Control unit
204
106
CRYPTO unit
108
Auth. Unit 1 HMAC 1
110
Auth. Unit 2 HMAC 2
112
206
Packet Processor

Fig. 2



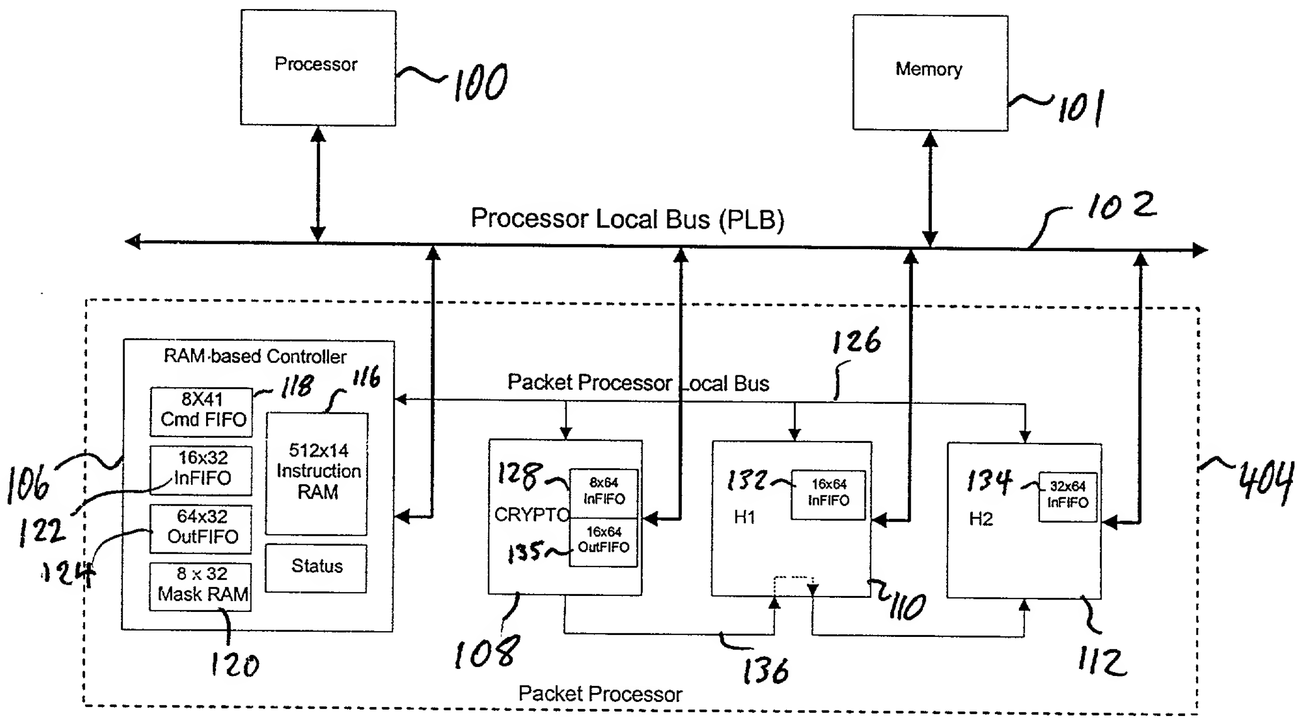
Computer/ processor
100
Memory
101
102
202
Controller Unit
CRYPTO Unit
Auth. Unit 1 HMAC 1
Auth. Unit 2 HMAC 2
110
112
304
106
108
206
Packet Processor

Fig. 3



CRYPTO Unit (108)
1st Packet
2nd Packet
3rd Packet
4th Packet
HMAC 1 Auth. Unit 1 (110)
1st Packet
2nd Packet
3rd Packet
4th Packet
HMAC 2 Auth. Unit 2 (112)
1st Packet
2nd Packet
3rd Packet
4th Packet

Fig. 4



Processor
100
Memory
101
Processor Local Bus (PLB)
102
RAM-based Controller
118
116
8X41 Cmd FIFO
16x32 InFIFO
64x32 OutFIFO
8 x 32 Mask RAM
512x14 Instruction RAM
Status
106
122
124
120
Packet Processor Local Bus
126
128
8x64 InFIFO
CRYPTO
135
16x64 OutFIFO
132
16x64 InFIFO
H1
134
32x64 InFIFO
H2
108
136
110
112
404
Packet Processor

Fig. 5

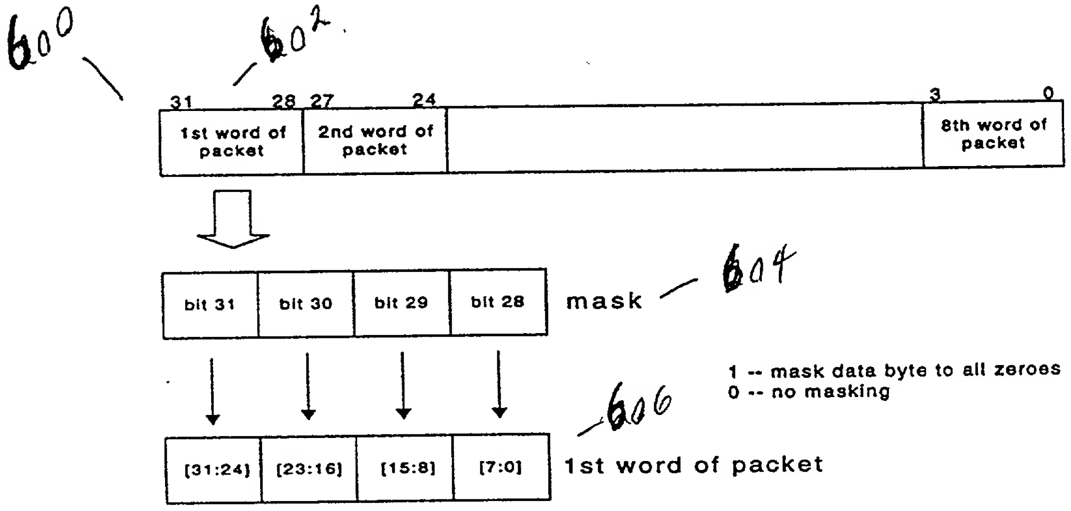
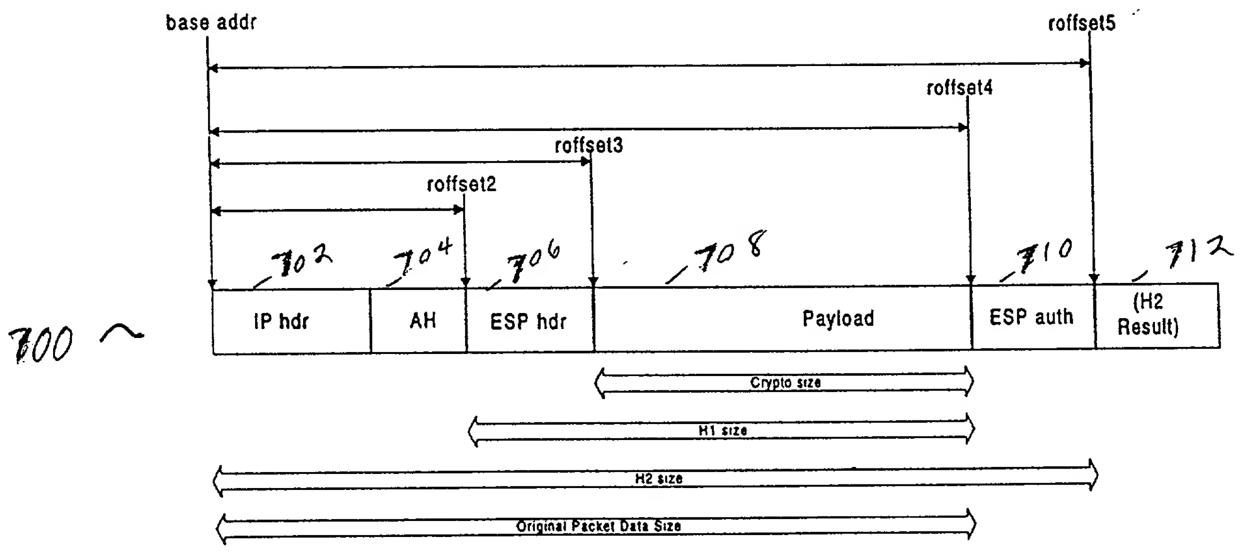

600
602
31
28
27
24
3
0
1st word of packet
2nd word of packet
8th word of packet
bit 31
bit 30
bit 29
bit 28
mask
604
1 -- mask data byte to all zeroes
0 -- no masking
606
[31:24]
[23:16]
[15:8]
[7:0]
1st word of packet

FIG. 6



base addr
roffset5
roffset4
roffset3
roffset2
702
704
706
708
710
712
700
IP hdr
AH
ESP hdr
Payload
ESP auth
(H2 Result)
Crypto size
H1 size
H2 size
Original Packet Data Size

FIG.7

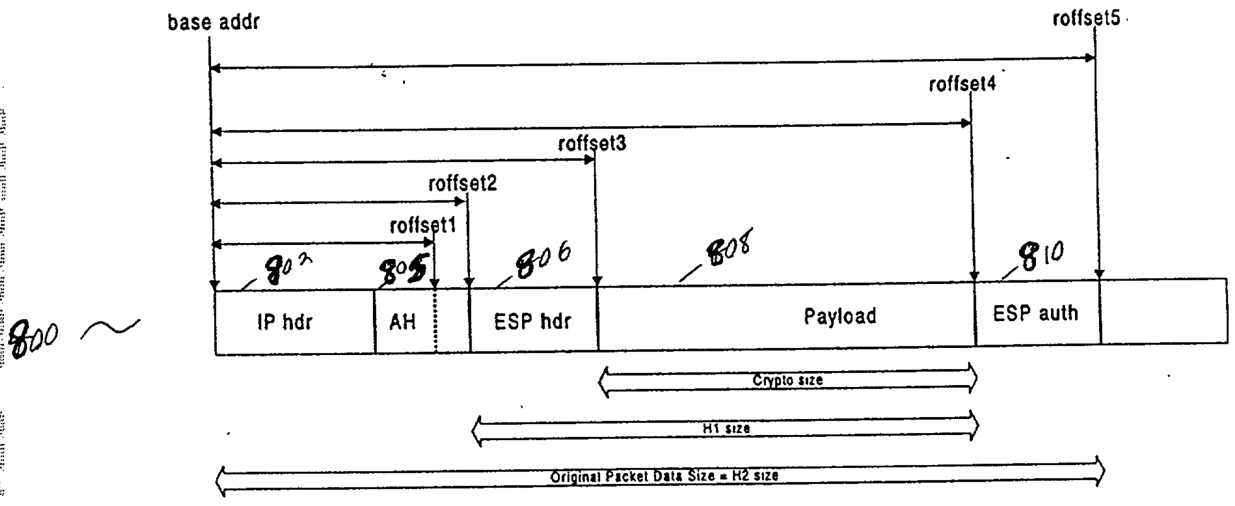

base addr
roffset5
roffset4
roffset3
roffset2
roffset1
802
805
806
808
810
800
IP hdr
AH
ESP hdr
Payload
ESP auth
Crypto size
H1 size
Original Packet Data Size = H2 size

FIG. 8

NetSwift ASIC Packet Processor

Table of Contents

List of Figures

List of Tables

Reference Section: Detailed Description of the Preferred Embodiment
(Page 7, lines29-30; Pages 8-36, all lines)

<u>The purpose of this application is to patent the Packet Processor bus architecture to be used for IPSec packet processing application</u>. This bus architecture is critical in providing the efficient data flow necessary for high throughput in processing IPSec packets.

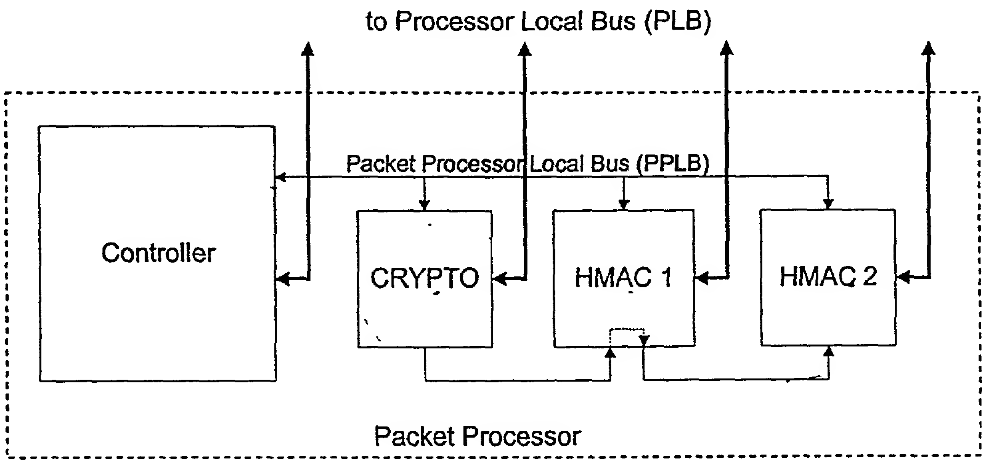

to Processor Local Bus (PLB)
Packet Processor Local Bus (PPLB)
Controller
CRYPTO
HMAC 1
HMAC 2
Packet Processor

Figure 1: Packet Processor Architecture

Figure 1 shows the internal architecture of the Packet Processor. The CRYPTO unit is used to encrypt and decrypt data. HMAC 1 and HMAC 2 are authentication units. They allow the Packet Processor to perform up to two authentication on a packet. The Controller regulates the packet processing operation and fetches the input packet data for the CRYPTO, HMAC 1, and HMAC 2 units when the Packet Processor is used.

Each of the units, the Controller, CRYPTO, HMAC 1, and HMAC 2, is capable of being a bus master (i.e., being able to take over the data bus to do its own read or write transfers) for both read and write operations on the shared data bus called the Processor Local Bus or PLB. These units can also act as slave devices and be accessed by the microprocessor through the PLB bus. These bus interfaces to the PLB bus allow the CRYPTO, HMAC 1, and HMAC 2 units to be used as independent units.

When the Packet Processor is enabled, the CRYPTO, HMAC 1, and HMAC 2 units become part of the Packet Processor. The packet processing data flow is shown in Figure 2. The Controller uses its bus-master interface to read in the processing parameters and input packet data. The Controller passes processing parameters on to the CRYPTO, HMAC 1, and/or HMAC 2 units via the Packet Processor internal bus called Packet Processor Local Bus or PPLB. Depending on whether the packet is inbound or outbound and the type of processing required, the Controller passes the input packet data on to the CRYPTO, HMAC 1, and/or HMAC 2 units via the PPLB bus in the appropriate order. For outbound packets where the input payload is plaint text, the cipher text output from the CRYPTO unit is transferred to the HMAC units for authentication via the daisy chain bus. Additionally, the HMAC 1 authentication value is made available to HMAC 2 for IPSec packets requiring two authentications. Thus, with this architecture, the input packet is read once into the Packet Processor to have up to three cryptographic operations performed on the packet.

1. Packet Processor Overview

The Packet Processor consists of a RAM-based controller, one Encryption (DES/3DES/RC4) unit, and two HMAC (HMAC/SHA-1/MD5) units. The Encryption unit and the two HMAC units can operate independently when the Packet Processor is not enabled. However, when the Packet Processor is enabled, the Encryption and HMAC units become part of the Packet Processor and receive inputs from the RAM-based Controller or from the daisy chain bus, as shown in Figure 1. The Packet Processor is disabled following power-up reset.

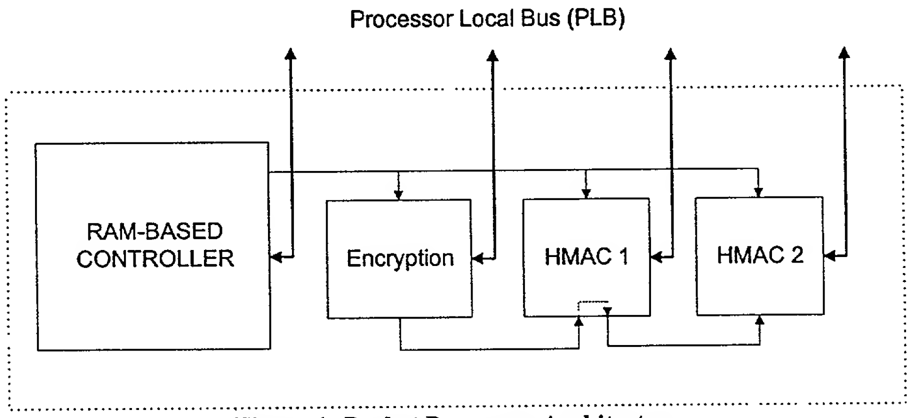


Figure 1: Packet Processor Architecture

2. RAM-based Controller

The RAM-based Controller is a programmable unit designed to control packet processing through the Encryption and HMAC units. The Controller has the following features:

- 16 dedicated internal registers
- an optimized instruction set for processing packet data
- Instruction RAM for storing up to 512 instructions
- 256-bit Mask RAM for masking out mutable fields in IP Authentication Header data
- Command FIFO for queuing up to 8 commands (one command for each packet to be processed)
- data transfer from/to memory using DMA, allowing the PPC to perform other tasks
- 16-word Input FIFO and 64-word Output FIFO
- 32-bit data bus for passing commands, parameters, and packet data to the Encryption and HMAC units
- packet processing status tagged at the end of packet output

2.1. Address Map

The RAM-based Controller dedicated registers and Mask RAM, along with registers from the Encryption and HMAC units, are address mapped to the internal Packet Processor 7-bit addressing space, as shown in Table 1. The RAM-based Controller registers and memory devices will be discussed in detail in the following section. Refer to the Encryption and HMAC sections for register definitions of those units.

Table 1: RAM-based Controller Internal Memory Map

PP Address Bus [6:0]	Device
0x00	Offset Register 0
0x01	Offset Register 1
0x02	Offset Register 2
0x03	Offset Register 3
0x04	Size Register 0
0x05	Size Register 1
0x06	Size Register 2
0x07	Size Register 3
0x08	Encryption Command Register
0x09	HMAC 1 Command Register
0x0A	HMAC 2 Command Register
0x0B	Packet Status Destination Address Register
0x0C	Offset Register 4
0x0D	Offset Register 5
0x0E	Offset Register 6
0x0F	Offset Register 7
0x15	Packet Command Register(PCR)
0x16	Base Address Register
0x17	Destination Address Register
0x18 – 0x1F	Mask RAM
0x21	Encryption RC4 Key Length Register
0x22	Encryption Key Register 0
0x23	Encryption Key Register 1

0x24	Encryption Key Register 2
0x25	Encryption Key Register 3
0x26	Encryption Key Register 4
0x27	Encryption Key Register 5
0x28	Encryption Key Register 6
0x29	Encryption Key Register 7
0x2A	Encryption DES IV Register 0
0x2B	Encryption DES IV Register 1
0x2C	Encryption Source Address Register
0x2D	Encryption Destination Address Register
0x2E	Encryption Command/Status Register
0x41	HMAC 1 Length Register 0
0x42	HMAC 1 Length Register 1
0x43	HMAC 1 Outer IV Register 0
0x44	HMAC 1 Outer IV Register 1
0x45	HMAC 1 Outer IV Register 2
0x46	HMAC 1 Outer IV Register 3
0x47	HMAC 1 Outer IV Register 4
0x48	HMAC 1 Hash Register 0
0x49	HMAC 1 Hash Register 1
0x4A	HMAC 1 Hash Register 2
0x4B	HMAC 1 Hash Register 3
0x4C	HMAC 1 Hash Register 4
0x4D	HMAC 1 Source Address Register
0x4E	HMAC 1 Destination Address Register
0x4F	HMAC 1 Command/Status Register
0x61	HMAC 2 Length Register 0
0x62	HMAC 2 Length Register 1
0x63	HMAC 2 Outer IV Register 0
0x64	HMAC 2 Outer IV Register 1
0x65	HMAC 2 Outer IV Register 2
0x66	HMAC 2 Outer IV Register 3
0x67	HMAC 2 Outer IV Register 4
0x68	HMAC 2 Hash Register 0
0x69	HMAC 2 Hash Register 1
0x6A	HMAC 2 Hash Register 2
0x6B	HMAC 2 Hash Register 3
0x6C	HMAC 2 Hash Register 4
0x6D	HMAC 2 Source Address Register
0x6E	HMAC 2 Destination Address Register
0x6F	HMAC 2 Command/Status Register

The RAM-based Controller is connected to the Processor Local Bus (PLB) as a slave device as well as a master device. The slave interface is for programming the RAM-based Controller Instruction RAM, commanding the Packet Processor, queuing up packets to be processed, and reading status. In addition, some of the RAM-based Controller internal registers may also be read via this interface. Table 2 and 3 show the PLB address map of these devices.

The RAM-based Controller is capable of becoming a master on the PLB bus, allowing it to DMA data into the Input FIFO and to DMA data out of the Output FIFO. When the master interface is not enabled, data is transferred to and from the RAM-based Controller via the slave interface.

Table 2: PLB Address Map for RAM-based Controller Registers

PLB Address	Register Name
0x20070044	Command/Status Register(CSR)
0x20070050	Configuration Register(CFG)
0x20070040	Instruction RAM Pointer Register
0x20070048	Packet Status Register (read only)
0x20070054	Packet Command Register (read only)
0x2007004C	Source Address Register (read only)
0x2007005C	Destination Address Register (read only)
0x2007002C	Packet Status Destination Address Register (read only)

Table 3: PLB Address Map for RAM-based Controller Memory Devices

PLB Address	Device Name	Size
0x20170000	Input FIFO	16 words
0x20170000	Output FIFO	64 words
0x20070060 – 0x2007007F	Mask RAM (0-7)	8 words
0x20070800 – 0x20070FFF	Instruction RAM	512 instructions
0x20171000 – 0x20171004	Command FIFO	8 commands

09505382 . 024402

2.2. Instruction Set

The RAM-based Controller processes packets according to instruction routines stored in the Instruction RAM. The instruction routines are to be constructed from the instruction set described in the following sections. The instructions are 14 bits wide; the four most significant bits specify the operational code and the remaining 10 bits specify the parameters, if any, associated with the instruction. All instructions operate on 32-bit words.

2.2.1. LOAD

Operational code: 0x0
Parameter: [number of words – 10 bits wide]

This instruction causes the RAM-based Controller to load the specified number of words from the location pointed to by the Source Address Register into its Input FIFO. If input DMA is disabled, no data transfer will occur.

2.2.2. STOREX

Operational code: 0x1
Parameters: [internal address – 7 bits wide][count – 3 bits wide]

This instruction causes the RAM-based Controller to store one or multiple words from its Input FIFO to devices located at addresses beginning from the specified [internal address]. The number of words transferred is [count] plus one.

Note that storing size to Size Register 1, 2, and 3 also triggers the following operations:

- Storing size to Size Register 1 also causes (I) Encryption end offset to be computed and stored in Offset Register 5 and (ii) Encryption destination address to be computed and stored in Encryption Destination Address Register. This requires that the Destination Address Register and Offset Register 1 be initialized before writing to this size register.
- Storing size to Size Register 2 also causes (I) HMAC 1 end offset to be computed and stored in Offset Register 6 and (ii) HMAC 1 destination address to be computed and stored in HMAC 1 Destination Address Register. This requires that the Destination Address Register and Offset Register 2 be initialized before writing to this size register.
- Storing size to Size Register 3 also causes (I) HMAC 2 end offset to be computed and stored in Offset Register 7 and (ii) HMAC 2 destination address to be computed and stored in HMAC 2 Destination Address Register. This requires that the Destination Address Register and Offset Register 3 be initialized before writing to this size register.

2.2.3. START_NEW_BASE

Operational code: 0x2
Parameter: none

This instruction causes the RAM-based Controller to load the content of the Base Address Register into the Source Address Register and to start transferring data into its Input FIFO. The number of words transferred is specified by Size Register 0. If input DMA is disabled, no data transfer will occur.

Execution of this instruction also causes the output interface to be enabled (if DMA is enabled) and the byte-masking logic to be initialized.

2.2.4. WRITE

Operational code: 0x3
Parameter: [function select – 10 bits wide]

Execution of this instruction causes the selected function(s) to occur. The [function select] field is bit decoded as follow:

Bit 9-5 – **Reserved.** These bits should be set to zero.

Bit 4 – **Enable HMAC 2 Daisy Chain Input.** This bit set commands HMAC 2 to receive the remaining input data from the daisy chain bus.

Bit 3 – **Enable HMAC 1 Daisy Chain Input.** This bit set commands HMAC 1 to receive the remaining input data from the daisy chain bus.

Bit 2 – **Reset Output FIFO.** This bit set commands the RAM-based Controller to reset its Output FIFO.

Bit 1 – **Write All.** This bit set commands the RAM-based Controller to write the remaining content of its Output FIFO to memory. This command should be used only once for each packet and should be executed after all inputs into the Output FIFO are loaded. Execution of this command is required if sequential output is enabled, even if the Output FIFO is empty.

Bit 0 – **Load Packet Status Destination Address Register.** This bit set causes the destination address for the packet status to be computed and loaded into the Packet Status Destination Address Register. This must be done before executing the WRITE_DATA instruction.

2.2.5. WRITE_STATBLKCMD

Operational code: 0x4
Parameter: none

This instruction causes the RAM-based Controller to set the appropriate enable bits in the Packet Status Register to enable status reporting for the appropriate units.

2.2.6. WRITE_DATA

Operational code: 0x5
Parameters: [write configuration – 7 bits wide][offset register select – 3 bits wide]

This instruction causes the RAM-based Controller to broadcast data from the RAM-based Controller Input FIFO to the destinations specified by the write configuration. The [write configuration] field is bit decoded as follow:

Bit 6 – **Reserved.** This bit should be set to zero.

Bit 5 – **HMAC 1 ICV Write Enable.** This bit set enables writing to the HMAC 1 Expected ICV Register.

Bit 4 – **HMAC 1 Input FIFO Write Enable.** This bit set enables writing to the HMAC 1 Input FIFO.

Bit 3 – **HMAC 2 ICV Write.** This bit set enables writing to the HMAC 2 Expected ICV Register.

Bit 2 – **HMAC 2 Input FIFO Write Enable.** This bit set enables writing to the HMAC 2 Input FIFO.

Bit 1 – **Encryption Input FIFO Write Enable.** This bit set enables writing to the Encryption Input FIFO.

Bit 0 – **RAM-based Controller Output FIFO Write Enable.** This bit set enables writing to the RAM-based Controller Output FIFO.

The number of words to be transferred is indicated by the selected Offset Register. "000" in the [offset register select] field selects Offset Register 0. "111" in the [offset register select] field selects Offset Register 7. If the destination is a FIFO, the RAM-based Controller checks the corresponding FIFO full flag before writing. Byte masking for HMAC 1 and HMAC 2, if enabled, takes effect while executing this instruction. All offset registers decrement while executing this instruction.

2.2.7. WAIT

Operational code: 0x6
Parameter: [condition – 10 bits wide]

This instruction causes the RAM-based Controller to wait until the specified conditions are met. Conditions are checked only if the corresponding units are enabled (via the Packet Command Register). The [condition] field is bit-decoded as follow:

Bit 9-8 – Reserved. These bits should be set to zero

Bit 7 – HMAC 1 not busy

Bit 6 – HMAC 1 buffer not full

Bit 5 – HMAC 1 Input FIFO available

Bit 4 – HMAC 2 not busy

Bit 3 – HMAC 2 buffer not full

Bit 2 – HMAC 2 Input FIFO available

Bit 1 – Encryption not busy

Bit 0 – RAM-based Controller Output FIFO empty

This instruction should be used to verify that the Encryption unit is not busy before performing any write to it; that the HMAC buffers are not full before writing any parameter or command to the units; and that the HMAC Input FIFOs are available before writing commands to the units and loading data to the FIFOs. These conditions (bits 1-7) need to be checked once before writing data from a new packet to the corresponding units.

2.2.8. STOP

Operational code: 0x7
Parameter: none

This instruction indicates the end of a service routine. After executing this instruction, the RAM-based Controller starts to process the next command from the Command FIFO, if applicable (depending on the control bits in the Command/Status Register and the Configuration Register).

2.3. Initialization

Several operations must occur before the Packet Processor may be used. The Instruction RAM must be initialized with instruction routines to be used to process the various IPSec packets. The Configuration Registers of the RAM-based Controller, Encryption unit, and HMAC units must be set up to the desired mode and PLB bus priority (if DMA transfer will be used). Depending on the required execution mode of the Packet Processor (refer to the definition of the Command/Status Register), the Command/Status Register may be initialized at this time (for Execute Until Stop mode) to turn "ON" the Packet Processor.

2.4. Packet Processing Operation

The Packet Processor processes IPSec packets through the use of packet control structures. A packet control structure is constructed either by the host or the PPC to specify how a packet is to be processed and to pass parameters. The following is a sample list of information that may be contained in the packet control structure:

- Input packet data address and output destination address.
- Starting/ending offset of data to be encrypted or authenticated.
- Size of data to be processed in each of the units.
- RC4, DES, 3DES, HMAC 1, and/or HMAC 2 keys.
- DES, 3DES, HMAC 1, and or HMAC 2 Initial Values.
- Algorithm, Mask data, processing modes.

The order of information and parameters in the packet control structure must correspond with the instruction routine to be used to process it and its packet data. The location of the packet control structure, together with the Instruction RAM offset of the routine, are loaded into the Command FIFO as a command for the Packet Processor. Up to eight packets may be queued in the Command FIFO for back-to-back packet processing.

When the Packet Processor is "ON", the RAM-based Controller unloads a command from the Command FIFO as soon as it finishes processing the current packet. The Instruction RAM offset is loaded to the Instruction RAM Pointer Register, and the address of the packet control structure is loaded to the Source Address Register. The RAM-based Controller then starts to execute the instruction routine pointed to by the Instruction RAM Pointer Register. Depending on the execution mode selected in the Command/Status Register, the RAM-based Controller can start processing the next command from the Command FIFO after it executes the STOP instruction of the current routine.

The Packet Processor finishes processing a packet when all of the units (Encryption, HMAC 1, and/or HMAC 2) that are required to process the packet have completed. This is indicated by the most significant bit of the Packet Status Register, which, when set, also signifies that other status bits in the register are valid for interpretation.

The Packet Processor can generate interrupts due to several conditions: (i) after each packet is processed and all outputs, including the status word, are written out to the destination, (ii) when the Command FIFO has the programmed number of spaces available, and (iii) when the Busy bit of the Command/Status Register transitions from one to zero. The Encryption and HMAC units also generate interrupts when the Busy bits of their Command/Status Registers transition from one to zero, indicating the completion of data processing. All of these critical interrupts are disabled following reset.

Note that if a packet requires utilization of more than one unit, the command registers of these units must be written in the order of Encryption Command/Status Register, followed by HMAC 1 Command/Status Register, and then HMAC 2 Command/Status Register.

2.5. Internal Calculations

The RAM-based Controller performs several calculations based on the information provided through the packet control structure. The following algorithm is used to compute the destination address for the packet status such that the packet status is appended to the end of the packet output.

For outbound packets,

```
If HMAC 2 is enabled, then
  Packet Status Destination Address = [Destination Address Register]+[Offset Register 7]+[HMAC 2 result size]
Else if HMAC 1 is enabled, then
  Packet Status Destination Address = [Destination Address Register]+[Size Register 0]+[HMAC 1 result size]
Else
  Packet Status Destination Address = [Destination Address Register]+[Size Register 0].
```

For inbound packets,

```
If HMAC 2 is enabled and HMAC out DMA is enabled, then
  Packet Status Destination Address = [Destination Address Register]+[Size Register 0]+[HMAC 2 result size]
Else if HMAC 1 is enabled and HMAC out DMA is enabled, then
  Packet Status Destination Address = [Destination Address Register]+[Size Register 0]+[HMAC 1 result size]
Else if Encryption is enabled, then
  Packet Status Destination Address = [Destination Address Register]+[Offset Register 5]
Else
  Packet Status Destination Address = [Destination Address Register]+[Size Register 0].
```

The address calculated above is used when the WRITE instruction is used to load the Packet Status Destination Address Register. The destination address may also be pre-computed and passed to the Packet Status Destination Address Register through the packet control structure by using the STOREX instruction.

Destination addresses for the Encryption, HMAC 1, and HMAC 2 units are computed as shown below and written to the Destination Address Registers of the units when Size Registers 1, 2, and 3 are loaded, respectively (refer to the description for STOREX instruction). These destination addresses may also be pre-computed and passed to these registers through the packet control structure by using the STOREX instruction with the appropriate internal addresses.

Encryption Destination Address	= [Destination Address Register]+[Offset Register 1]
HMAC 1 Destination Address	= [Destination Address Register]+[Offset Register 2]+[Size Register 2].
HMAC 2 Destination Address	= [Destination Address Register]+[Offset Register 3]+[Size Register 3].

2.6. Register Definitions

All registers internal to the RAM-based Controller are 32 bits wide, except for the Offset and Size Registers, which are 16 bits wide.

2.6.1. Configuration Register

PLB Address: 0x20070050

The bit-definition of this register is shown in Table 4. Writing to this register is allowed only when the Busy bit of the Command/Status Register is clear, except for the Terminate and Stop bits, which may be updated at any time.

Table 4: Configuration Register

Bit #	Description
31	**Terminate.** This bit set commands the Packet Processor to stop immediately. The PPC also needs to set the Terminate bits of the Encryption and HMAC units as well. This bit should be set to zero during normal operation. This bit may be updated at any time.
30	**Stop When Empty.** This bit set commands the Packet Processor to stop when all commands in the Command FIFO are processed. This bit is used only when the Execute until Stop bit of the Command/Status Register is set and DMA is enabled. This bit can be updated at any time.
29	**Stop When Done.** This bit set commands the Packet Processor to stop after it finishes processing the current command. This bit is used only when the Execute until Stop bit of the Command/Status Register is set and DMA is enabled. This bit can be updated at any time.
28:16	**Reserved.** These bits should be set to all zeroes.
15:12	**Command FIFO Interrupt Threshold.** These bits specify when, in term of the number of spaces available in the Command FIFO, to generate an interrupt. Valid values are 0 through 8.
11:9	**Reserved.** These bits should be set to all zeroes.
8	**HMAC Output DMA Enable.** This bit set specifies to always write out HMAC 1 and HMAC 2 results. This bit clear specifies to write out HMAC 1 and HMAC 2 results only when it is an outbound packet and the Output DMA Enable bit of the Command/Status Register is set.
7	**Packet Processor Enable.** This bit set enables the Packet Processor to control the Encryption and HMAC units. This bit clear disables the Packet Processor, and thus, the Encryption and HMAC units operate independent from each other and are controlled by the PPC.

6	**Instruction RAM Configuration Enable.** This bit set allows the PPC to write to the Instruction RAM. This bit clear inhibits all writing to the Instruction RAM.
5	**Reset Command FIFO.** This bit set specifies to reset the Command FIFO. This bit should be clear during normal operation.
4	**Sequential Output Disable.** This bit clear commands the Packet Processor to write out packet results sequentially. This bit set commands the Packet Processor to write out results as they become available (not necessarily sequential).
3:2	**Status Local Bus Priority.** These bits specify the priority to be used when writing out the packet status using the PLB master interface. 00 specify lowest priority; 11 specify highest priority.
1:0	**FIFO Local Bus Priority.** These bits specify the PLB bus priority to be used when performing DMA transfer of data to/from the RAM-based Controller Input and Output FIFO using the master interface. 00 specify lowest priority; 11 specify highest priority.

2.6.2. Command/Status Register

PLB Address: 0x20070044

The bit-definition of this register is shown in table 5. Writing to this register turns "ON" the Packet Processor and sets the Busy bit. Writing to this register is not allowed when the Busy bit is set.

Table 5: Command/Status Register

Bit #	Description
31	**Busy.** This read-only bit set indicates that the Packet Processor is "ON", i.e., either is idle waiting for more packets or is processing packets. This bit clear indicates that the packet processor is "OFF".
30	**Command FIFO Full.** This read-only bit set indicates the Command FIFO is full.
29	**Command FIFO Empty.** This read-only bit set indicates the Command FIFO is empty.
28	**HMAC 1 Busy.** This read-only bit set indicates that HMAC 1 is busy. This bit clear indicates that HMAC 1 is idle.
27:26	**Reserved.**
25	**HMAC 2 Busy.** This read-only bit set indicates that HMAC 2 is busy. This bit clear indicates that HMAC 2 is idle.
24:23	**Reserved.**
22	**Encryption Busy.** This read-only bit set indicates that Encryption is busy. This bit clear indicates that Encryption is idle.
21	**Controller Output FIFO Empty.** This read-only bit set indicates that the RAM-based Controller Output FIFO is empty. This bit clear indicates that the FIFO is not empty.
20:6	**Reserved.**
5:4	**Execution Mode.** These bits specify how the Packet Processor should process commands from the Command FIFO and are decoded as follow: **00 – Execute Until Stop.** The Packet Processor processes commands from the Command FIFO until one of the Stop bits in the Configuration Register is set and then clears the Busy bit.

	01 – Execute Until Empty. The Packet Processor processes commands from the Command FIFO until the Command FIFO becomes empty and then clears the Busy bit. **1x – Execute One Command.** The Packet Processor executes one command from the Command FIFO and then clears the Busy bit. These bits are ignored if Input DMA is disabled.
3	**Output DMA AutoIncrement Disable.** This bit clear specifies to increment the destination address when using the PLB master interfaces to write out packet processing results. This bit set specifies not to increment the destination address.
2	**Output DMA Enable.** This bit set enables the Packet Processor to write out the results using the PLB master interfaces. This bit clear disables all PLB master write interfaces.
1	**Input DMA AutoIncrement Disable.** This bit clear specifies to increment the source address when using the PLB master interface to read in data. This bit set specifies not to increment the source address.
0	**Input DMA Enable.** This bit set enables the Packet Processor to read in data using the PLB master interface. If this bit is clear, the PPC must initializes the Instruction RAM Pointer Register, write to this register, and then loads the packet control structure and packet data to the RAM-based Controller Input FIFO for each packet.

2.6.3. Packet Command Register

PP Address: 0x15
PLB Address: 0x20070054

The Packet Command Register is used to specify how the packet should be processed. The bit definitions are shown in Table 6. This register is loaded from the Input FIFO by using the STOREX instruction. This register should be initialized first. This register may be read using the PLB address above.

Table 6: Packet Command Register

Bit #	Description
31	**HMAC 1 Mask Enable** – This bit set enables byte masking of data to be written to HMAC 1 Input FIFO. This bit clear specifies not to mask data.
30:28	**HMAC 1 Mask Size** – These bits select the number of 32-bit mask words to be applied. "000" enables byte masking of the first 32 bytes of packet data. "001" enables byte masking of the first 64 bytes of packet data. "111" enables byte masking of the first 256 bytes of packet data.
27	**HMAC 1 Initialize Hash.** This bit set specifies to use the default initial value specified by the algorithm as the starting hash value. This bit clear specifies to use the value currently in the Hash Registers as the starting hash value.
26	**HMAC 1 Final Block.** This bit set specifies to append padding and complete the hash operation. If the HMAC algorithm is selected, the unit will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding or length should be appended. Note that size must be multiples of 512 bits if this bit is not set.
25:24	**HMAC 1 Algorithm.** These bits specify the algorithm to be performed and are decoded as follow: 00 -- MD5 01 -- SHA-1 10 -- HMAC-MD5

	11 -- HMAC-SHA-1
23	**HMAC 2 Mask Enable** – This bit set enables byte masking of data to be written to HMAC 2 Input FIFO. This bit clear specifies not to mask data.
22:20	**HMAC 2 Mask Size** – These bits select the number of 32-bit mask words to be applied. "000" enables byte masking of the first 32 bytes of packet data. "001" enables byte masking of the first 64 bytes of packet data. "111" enables byte masking of the first 256 bytes of packet data.
19	**HMAC 2 Initialize Hash.** This bit set specifies to use the default initial value specified by the algorithm as the starting hash value. This bit clear specifies to use the value currently in the Hash Registers as the starting hash value.
18	**HMAC 2 Final Block.** This bit set specifies to append padding and complete the hash operation. If the HMAC algorithm is selected, the unit will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding or length should be appended. Note that size must be multiples of 512 bits if this bit is not set.
17:16	**HMAC 2 Algorithm.** These bits specify the algorithm to be performed and are decoded as follow: 00 -- MD5 01 -- SHA-1 10 -- HMAC-MD5 11 -- HMAC-SHA-1
15	**HMAC 1 Length/IPAD/OPAD Select.** When the Initialize Hash bit is set and the Final Block bit is clear, this bit is used to select between HMAC IPAD and OPAD. This bit set specifies to use the HMAC IPAD; this bit clear specifies to use the HMAC OPAD. This bit is used when performing the HMAC Inner and Outer IV generation commands. Otherwise, this bit is used to select the source of the message length. This bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use the Size field as the length of the message.
14	**HMAC 2 Length/IPAD/OPAD Select.** When the Initialize Hash bit is set and the Final Block bit is clear, this bit is used to select between HMAC IPAD and OPAD. This bit set specifies to use the HMAC IPAD; this bit clear specifies to use the HMAC OPAD. This bit is used when performing the HMAC Inner and Outer IV generation commands. Otherwise, this bit is used to select the source of the message length. This bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use the Size field as the length of the message.
13	**HMAC using HMAC 1 and HMAC 2.** This bit set specifies to use both HMAC units to perform the HMAC Final command. Note that HMAC 2 offsets and size should be the same as HMAC 1 offsets and size when performing this operation. If this is an inbound packet, the expected authentication value should be loaded to the HMAC 2 ICV Registers. This bit clear specifies to perform the entire HMAC operation using one HMAC unit.
12	**Encryption 3DES Keys for Decryption/Initialize RC4.** When Encryption unit is set to perform a 3DES operation, this bit specifies whether the 3DES key is for encryption or decryption. This bit set specifies that keys 1,2, and 3 are in the order for decryption and that the order of the keys should be reversed if encryption mode is selected. This bit clear specifies that keys 1,2, and 3 are in the order for encryption and that the order of the keys should be reversed if decryption mode is selected. Otherwise, this bit specifies whether RC4 initialization should be performed. This bit set specifies to initialize the RC4 engine with the key loaded in the Encryption Key Registers. This bit clear

	specifies to use the key stream that is currently in the RC4 engine.

11:8	**Encryption Algorithm[3:0]**. These bits specify the algorithm to be performed and are decoded as follow:
	bit 3: 1= RC4 0=DES/3DES;
	bit 2: 1= 3DES 0= DES;
	bit 1: 1= ECB 0= CBC;
	bit 0: 1= decryption 0= encryption

7	**Reserved.** This bit should be set to zero.
6	**Encryption Enable.** This bit set indicates that the Encryption unit is required for the processing of this packet. This bit clear indicates that the Encryption unit is not required for the processing of this packet.
5	**HMAC 1 Enable.** This bit set indicates that the HMAC 1 unit is required for the processing of this packet. This bit clear indicates that the HMAC 1 unit is not required for the processing of this packet.
4	**HMAC 2 Enable.** This bit set indicates that the HMAC 2 unit is required for the processing of this packet. This bit clear indicates that the HMAC 2 unit is not required for the processing of this packet.
3:2	**Mode Select.** These bits specify the size of the authentication value and the selection is applied to both HMAC units. 00 – selects to use only the leftmost 96 bits. 01 – selects to use only the leftmost 128 bits. 1x – selects to use the entire authentication value (160 bits for SHA-1, 128 bits for MD5).
1	**Inbound Packet.** This bit set indicates that this is an inbound packet. This bit clear indicates that this is an outbound packet.
0	**Endian.** This bit set specifies that the packet data to be processed is in little endian format. This bit clear specifies that the packet data to be processed is in big endian format. This bit also specifies the endian format for outputs from the packet processor, except for the packet status.

2.6.4. Base Address Register

PP Address: 0x016

This register specifies the source address of the input packet data to be processed. This register is loaded from the Input FIFO by using the STOREX instruction.

2.6.5. Destination Address Register

PP Address: 0x017
PLB Address: 0x2007005C

This register specifies the destination address where the Packet Processor output will be written. This register is loaded from the Input FIFO by using the STOREX instruction. This register may be read using the PLB address above.

2.6.6. Offset Registers 0-7

PP Addresses: 0x00, 0x01, 0x02, 0x03, 0x0C, 0x0D, 0x0E, 0x0F

An input packet may compose of an IP header, an AH header, an ESP header, a payload, authentication data, etc... Since the various segments in a packet may not be processed the same way, eight Offset Registers are provided for specifying the offsets of the starting or ending of certain data segment. Offset Registers 1, 2, and 3 may be used to store the starting offset for data segments to be processed in the Encryption, HMAC1, and HMAC2 units, respectively. Similarly, Offset Registers 5, 6, and 7 may be used for the ending offsets.

The Offset Registers are 16 bits wide and indicate the number of bytes from the beginning of a packet. They are loaded from the 16 least significant bits of the Input FIFO by using the STOREX instruction. The Offset Registers 5, 6, and 7 are also loaded automatically with the internally computed values when Size Registers 1, 2, and 3 are loaded, respectively. Refer to the STOREX instruction description for more details. All non-zero Offset Registers decrement when the WRITE_DATA instruction is executed. Note that since all data are on 32-bit boundary, the two least significant bits are ignored.

2.6.7. Size Registers 0-3

PP Addresses: 0x04, 0x05, 0x06, 0x07

Size Register 0 specifies the total number of bytes in the packet data, while Size Registers 1-3 specify the size of the data to be processed in the Encryption, HMAC 1, and HMAC 2 units, respectively.

The Size Registers are 16 bits wide and are loaded from the 16 least significant bits of the Input FIFO by using the STOREX instruction. Since data is multiples of 32-bit words, the two least significant bits are ignored.

2.6.8. Encryption Command Register

PP Address: 0x08

The Encryption Command Register is the command register in the Encryption unit, also known as the Encryption Command/Status Register at PP address 0x2E. When this register is loaded using the PP address 0x08, the command is derived from the information specified in the RAM-based Controller Packet Command Register, Command/Status Register, Configuration Register, and the Size Registers.

2.6.9. HMAC 1 Command Register

PP Address: 0x09

The HMAC 1 Command Register is the command register in the HMAC 1 unit, also known as HMAC 1 Command/Status Register at PP address 0x4F. When this register is loaded using the PP address 0x09, the command is derived from the information specified in the RAM-based Controller Packet Command Register, Command/Status Register, Configuration Register, and the Size Registers.

2.6.10. HMAC 2 Command Register

PP Address: 0x0A

The HMAC 2 Command Register is the command register in the HMAC 2 unit, also known as the HMAC 2 Command/Status Register at PP address 0x6F. When this register is loaded using the PP address 0x0A, the command is derived from the information specified in the RAM-based Controller Packet Command Register, Command/Status Register, Configuration Register, and the Size Registers.

2.6.11. Packet Status Destination Address Register

PP Address: 0x0B
PLB Address: 0x2007002C

This register stores the destination address for the status that is generated at the end of a packet processing. This register is loaded from the Input FIFO by using the STOREX instruction or from the internally computed address by using the WRITE instruction. This register may be read using the PLB address above.

2.6.12. Packet Status Register

PLB Address: 0x20070048

This read-only register reports the status of the packet being processed. The enable bits are set by the RAM-based Controller, and the status bits are set as the conditions become true. The most significant bit is set when processing of the packet has completed. If output DMA is enabled, the content of this register is written out to the address specified in the Packet Status Destination Address Register. The bit-definition of this register is shown in Table 7.

Table 7: Packet Status Register

Bit #	Description
31	**Completed packet processing.** This bit set indicates that packet processing has completed and all status bits in this register are valid. This bit clear indicates that packet processing has not completed and status bits in this register should be ignored.
30:24	**Reserved.** Read as zero.
23	**Encryption Status Enable.** This bit set specifies to report status for the Encryption unit. This bit clear specifies not to report status for the Encryption unit.
22	**Encryption Done.** This bit set indicates that the Encryption unit has completed processing. This bit clear indicates that the Encryption unit has not completed processing. This bit should be ignored when the Encryption Status Enable bit is low.
21	**Encryption Key error.** When the Encryption unit is set to perform an RC4 operation, this bit set indicates that the key length is 0 or greater than 32 bytes. This bit clear indicates that the key length is valid. Otherwise, this bit set indicates that the keys (key 1 for DES; keys 1, 2, and 3 for 3DES) contain parity error. This bit clear indicates that each byte of the keys has odd parity. Note that key errors do not stop the packet from being processed. This bit should be ignored when the Encryption Status Enable bit is low.
20:16	**Reserved.** Read as zero.
15	**HMAC 2 Status Enable.** This bit set specifies to report status for the HMAC 2 unit. This bit clear specifies not to report status for the HMAC 2 unit.
14	**HMAC 2 Done.** This bit set indicates that the HMAC 2 unit has completed processing. This bit clear indicates that the HMAC 2 unit has not completed processing. This bit should be ignored when the HMAC 2 Status Enable bit is low.
13	**HMAC 2 Size Error.** This bit set indicates that the data size is not an even multiple of 64 bytes when performing the HMAC/hash update command and that the result should be discarded. This bit clear indicates that the data size is valid. This bit should be ignored when the HMAC 2 Status Enable bit is low.
12	**HMAC 2 ICV Check Enable.** This bit set specifies to compare HMAC 2 result with the expected authentication value from the inbound packet.
11	**HMAC 2 ICV Check Status.** This bit set indicates that the HMAC 2 result matches with the expected authentication value. This bit clear indicates that the comparison failed. This bit should be ignored when the HMAC 2 ICV Check Enable bit is low.

10:8	**Reserved.** Read as zero.
7	**HMAC 1 Status Enable.** This bit set specifies to report status for the HMAC 1 unit. This bit clear specifies not to report status for the HMAC 1 unit.
6	**HMAC 1 Done.** This bit set indicates that the HMAC 1 unit has completed processing. This bit clear indicates that the HMAC 1 unit has not completed processing. This bit should be ignored when the HMAC 1 Status Enable bit is low.
5	**HMAC 1 Size Error.** This bit set indicates that the data size is not an even multiple of 64 bytes when performing the HMAC/hash update command and that the result should be discarded. This bit clear indicates that the data size is valid. This bit should be ignored when the HMAC 1 Status Enable bit is low.
4	**HMAC 1 ICV Check Enable.** This bit set specifies to compare HMAC 1 result with the expected authentication value from the inbound packet.
3	**HMAC 1 ICV Check Status.** This bit set indicates that the HMAC 1 result matches with the expected authentication value. This bit clear indicates that the comparison failed. This bit should be ignored when the HMAC 1 ICV Check Enable bit is low.
2 : 0	**Reserved.** Read as zero.

2.6.13. Instruction RAM Pointer Register

PLB Address: 0x20070040

This 9-bit register stores the pointer to the instruction routine in the Instruction RAM to be used to process the packet. This register loads from the nine least significant bits of the PLB data bus and advances as the RAM-based Controller executes the instructions.

This register is initialized by the PPC only when the input DMA is disabled. Otherwise, this register is loaded by the RAM-based Controller with data from the Command FIFO.

2.6.14. Source Address Register

PLB Address: 0x2007004C

This register is used internal to the RAM-based Controller to buffer the source address for the input packet control structure and data. This register may be read using the PLB address above.

2.7. Mask RAM

PP Addresses: 0x18 – 0x1F
PLB Address: 0x20070060 – 0x2007007F

The Mask RAM stores mask data for the first 256 bytes of packet data. The RAM may be loaded either by the PPC via the PLB slave bus or by the RAM-based Controller via the STOREX instruction. The application of the mask data to HMAC 1 and/or HMAC 2 Input FIFO data is controlled by information in the Packet Command Register.

The Mask RAM is 8 deep and 32 bits wide. The format of the Mask RAM is shown in Figure 2 below (for the first Mask RAM location). The masking data is for packet data in big endian format. If the packet data is in little endian format, the RAM-based Controller will swap the bits.

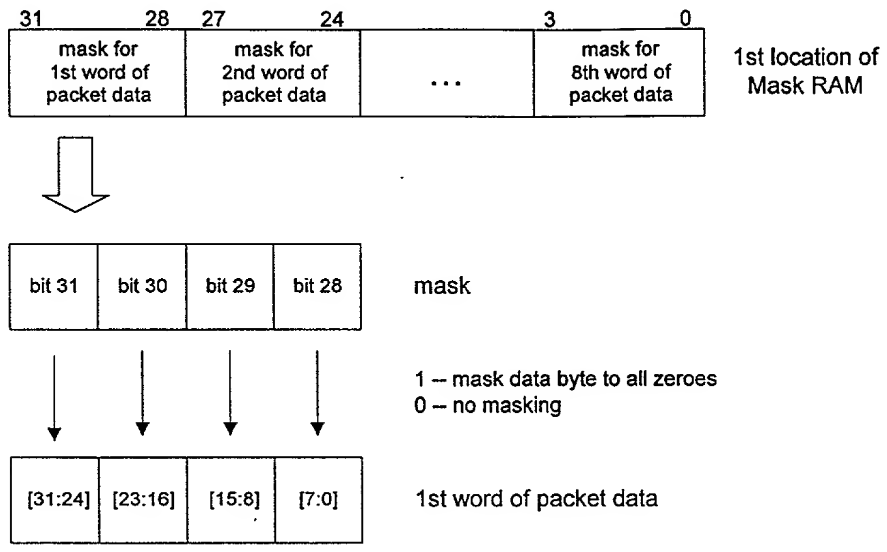


Figure 2: Mask RAM Format

2.8. Instruction RAM

PLB Address: 0x20070800 – 0x20070FFF

Instruction routines constructed from the instruction set are stored in the Instruction RAM. The Instruction RAM can store up to 512 instructions and must be initialized before the Packet Processor is used. The Instruction RAM loads from the lower 14 bits of the PLB slave bus when writing to the PLB address above while the Instruction RAM Configuration Enable bit of the Configuration Register is set.

2.9. Command FIFO

PLB Addresses: 0x20171000 – 0x20171004

For each packet to be processed by the Packet Processor, the PPC must enter a command into the Command FIFO via the PLB slave interface to specify the instruction routine to be used and the location of the packet control structure. The Instruction RAM offset is temporarily stored in a register at PLB address 0x20171000. When the address of the packet control structure is written to PLB address 0x20171004, the entire command is entered into the Command FIFO. Reading the Command FIFO via the PLB slave interface returns the next command to be processed (without unloading the FIFO).

The Command FIFO is unloaded by the RAM-based Controller as soon as the Controller finishes processing the current packet. The Instruction RAM offset is loaded to the Instruction RAM Pointer Register and the address of the packet control structure is loaded to the Source Address Register. The RAM-based Controller then starts to execute the instruction routine pointed to by the Instruction RAM Pointer Register.

The Command FIFO can store up to eight commands. Its status flags can be read via the Command/Status Register. The Command FIFO can also generate an interrupt when the number of spaces available equals the threshold set in the Configuration Register. Resetting the Command FIFO may be done by setting bit 5 of the Configuration Register.

2.10. Input and Output FIFO

PLB Address: 0x20170000

The RAM-based Controller Input FIFO and Output FIFO may be accessed using the PLB address above. Data may also be transferred to/from the FIFOs using the PLB master interfaces, when DMA is enabled. Data is written into the Input FIFO. Data is read from the Output FIFO.

3. Encryption Unit

The Encryption unit supports DES in CBC and ECB modes, triple DES (3DES) in outer CBC and ECB modes, and RC4. The unit can encrypt at the rate of 528 Mbps for DES, 176 Mbps for 3DES, and 528 Mbps for RC4. Table 8 shows the encryption commands that are available from this unit.

Table 8: Encryption Unit Commands

Command	Description
DES CBC Complete	Perform complete DES CBC encryption or decryption.
DES ECB Complete	Perform complete DES ECB encryption or decryption.
3DES CBC Complete	Perform complete 3DES outer CBC encryption or decryption.
3DES ECB Complete	Perform complete 3DES ECB encryption or decryption.
RC4 Complete	Perform complete RC4 encryption or decryption.
RC4 Without Initialization	Perform RC4 encryption or decryption without initialization (continue from the previous processing).

The Encryption unit is connected to the Processor Local Bus (PLB) as a slave device as well as a master device. The slave interface may be used to access all registers, the 16-word Input FIFO, and the 32-word Output FIFO internal to the unit, while the master interface provides for the more efficient DMA transfer of data to/from the Input and Output FIFOs. When the Packet Processor is enabled, the Encryption unit accepts inputs only from the RAM-based Controller, except for the Encryption Configuration Register.

The Encryption unit starts processing data following a write to the Encryption Command/Status Register. While the unit is busy encrypting or decrypting, the Busy bit of the Encryption Command/Status Register is set to one. When processing completes, the Busy bit transitions from one to zero and a maskable critical interrupt is generated.

3.1. Address Map

The Encryption unit registers and memory devices are memory-mapped to the PLB address bus as shown in Table 9. All of the Encryption unit registers, except for the Encryption Configuration Register, are also memory mapped to the Packet Processor address bus, as shown in Table 1.

Table 9: PLB Address Map for the Encryption Unit

PLB Address	Device
0x20060200	Encryption Configuration Register
0x20060204	Encryption RC4 Key Length Register
0x20060208	Encryption Key Register 0
0x2006020C	Encryption Key Register 1
0x20060210	Encryption Key Register 2
0x20060214	Encryption Key Register 3
0x20060218	Encryption Key Register 4

0x2006021C	Encryption Key Register 5
0x20060220	Encryption Key Register 6
0x20060224	Encryption Key Register 7
0x20060228	Encryption DES IV Register 0
0x2006022C	Encryption DES IV Register 1
0x20060230	Encryption Source Address Register
0x20060234	Encryption Destination Address Register
0x20060238	Encryption Command/Status Register
0x20160000	Encryption Input FIFO
0x20160000	Encryption Output FIFO

3.2. Register Definitions

All registers internal to the Encryption unit are 32 bits wide, unless specified otherwise.

3.2.1. Encryption Configuration Register

PLB Address: 0x20060200

The bit-definition of this register is shown in table 10. This register may be updated at any time.

Table 10: Encryption Configuration Register

Bit #	Description
26:25	**Local Bus Priority.** These bits specify the PLB bus priority to be used when performing DMA transfer using the master interface. 00 specify lowest priority; 11 specify highest priority.
24	**Terminate.** This bit set commands the Encryption unit to stop immediately. This bit should be set to zero during normal operation.

3.2.2. Encryption Command/Status Register

PP Address: 0x2E
PLB Address: 0x20060238

The bit-definition of this register is shown in Table 11. Writing to this register starts the Encryption unit and sets the Busy bit. Writing to this register is inhibited while the Busy bit is set. The completion of processing is indicated by the transition of the Busy bit from one to zero.

Table 11: Encryption Command/Status Register

Bit #	Description
31	**Busy.** This read-only bit set indicates that the Encryption unit is busy processing data. This bit clear indicates that the Encryption unit is idle and is ready for a new command.
30	**RC4 Initialization Busy.** This read-only bit set indicates that RC4 is initializing. This bit clear indicates that RC4 initialization is not in progress.
29	**Encryption Key error.**

	When the Encryption unit is set to perform an RC4 operation, this bit set indicates that the key length is 0 or greater than 32 bytes. This bit clear indicates that the key length is valid. Processing will continue to completion, even though the resulting text should be discarded. Otherwise, this bit set indicates that the keys (key 1 for DES; keys 1, 2, and 3 for 3DES) contain parity error. This bit clear indicates that each byte of the keys has odd parity. This error does not stop the data from being processed.
28	**Output-To-HMAC 1 Enable.** This bit set commands the Encryption unit to pass the resulting text to HMAC 1.
27	**Output-To-HMAC 2 Enable.** This bit set commands the Encryption unit to pass the resulting text to HMAC 2.
26	**Output DMA AutoIncrement Disable.** This bit clear specifies to increment the destination address when using the PLB master interface to write out the results. This bit set specifies not to increment the destination address.
25	**Output DMA Enable.** This bit set specifies to write out the results using the PLB master interface. This bit clear disables the PLB master write interface.
24	**3DES Keys for Decryption.** This bit set specifies that keys 1,2, and 3 are in the order for decryption and that the order of the keys should be reversed if encryption mode is selected. This bit clear specifies that keys 1,2, and 3 are in the order for encryption and that the order of the keys should be reversed if decryption mode is selected.

23:20	**Encryption Algorithm[3:0].** These bits specify the algorithm to be performed and are decoded as follow:
	bit 3: 1= RC4 0=DES/3DES;
	bit 2: 1= 3DES 0= DES;
	bit 1: 1= ECB 0= CBC;
	bit 0: 1= decryption 0= encryption

19	**Initialize RC4.** This bit set specifies to initialize the RC4 engine with the key loaded in the Encryption Key Registers. This bit clear specifies to use the key stream that is currently in the RC4 engine.
18	**Input DMA AutoIncrement Disable.** This bit clear specifies to increment the source address when using the PLB master interface to read in data. This bit set specifies not to increment the source address.
17	**Input DMA Enable.** This bit set specifies to read in data using the PLB master interface. This bit clear disables the PLB master read interface.
16	**Endian.** This bit set specifies that the data is in little endian format. This bit clear specifies that the data is in big endian format.
15:0	Size. These bits specify the number of bytes to be processed. This field is initialized when this register is updated. During processing, this field is updated to reflect the number of bytes remaining to be processed. For DES and 3DES, data size must be an integral of 64-bit words.

3.2.3. Encryption Source Address Register

PP Address: 0x2C
PLB Address: 0x20060230

This 32-bit register stores the source address of the Encryption unit input data. This register is used when input DMA is enabled and the Packet Processor is disabled.

3.2.4. Encryption Destination Address Register

PP Address: 0x2D
PLB Addresses: 0x20060234

This 32-bit register stores the destination address for the Encryption unit output data. This register is used when output DMA is enabled.

3.2.5. Encryption RC4 Key Length Register

PP Address: 0x21
PLB Address: 0x20060204

This 6-bit register stores the RC4 key length in number of bytes. The maximum key length supported is 32 bytes. Register access is not allowed while the RC4 Initialization Busy bit is high.

3.2.6. Encryption Key Registers 0-7

PP Addresses: 0x22 – 0x29
PLB Addresses: 0x20060208 – 0x20060224

These eight 32-bit registers store the key for RC4, DES, and 3DES encryption. The Encryption Key Register 0 holds the most significant word of the key string, with the leftmost character of the key in the 31:24 bit position. Register access is not allowed when the RC4 Initialization Busy bit is high.

For DES and 3DES, key 1 is stored in Encryption Key Registers 0 and 1, key 2 is stored in Encryption Key Registers 2 and 3, and key 3 is stored in Encryption Key Registers 4 and 5. Each 64-bit key is loaded into the DES/3DES engine when the register with the higher address is updated.

3.2.7. Encryption DES IV Registers 0-1

PP Addresses: 0x2A, 0x2B
PLB Addresses: 0x20060228, 0x2006022C

These two 32-bit registers store the Initialization Vector for DES and 3DES in CBC mode. The Encryption DES IV Register 0 holds the most significant 32-bit word, with the leftmost character of the vector in the 31:24 bit position.

3.3. Encryption Input and Output FIFO

PLB Address: 0x20160000

The Encryption Input FIFO and Output FIFO may be accessed using the PLB address above. Data may also be transferred to/from the FIFOs using the PLB master interface, when DMA is enabled. When the Packet Processor is used, the Input FIFO accepts data only from the RAM-based Controller, through the use of the WRITE_DATA instruction. Data is written into the Input FIFO. Data is read from the Output FIFO.

4.5 HMAC 1 and HMAC 2

Two HMAC units are provided to authenticate ESP (Encapsulating Security Payload) and AH (Authentication Header) data in a single pass through the Packet Processor. The units are basically the same, with a few differences, and are called HMAC 1 and HMAC2. The HMAC 1 and HMAC 2 units support keyed-hashing for message authentication and hash algorithms SHA-1 and MD5. Each of the units can hash at the rate of 417 Mbps for SHA-1 and 519 Mbps for MD5.

The commands available from the HMAC units are shown in Table 12. Commands are provided to perform the HMAC operation in one or several steps. Figure 3 shows how the operation is divided. Depending on the application, a performance improvement may be achieved by storing the intermediate Inner and Outer IV and then using them for the next message that utilizes the same key. Performance may be improved further when the packet requires only one HMAC operation. Provision is made to perform the HMAC Final command using both HMAC 1 and HMAC 2 units, one to complete the inner hash operation and the other to complete the outer hash operation.

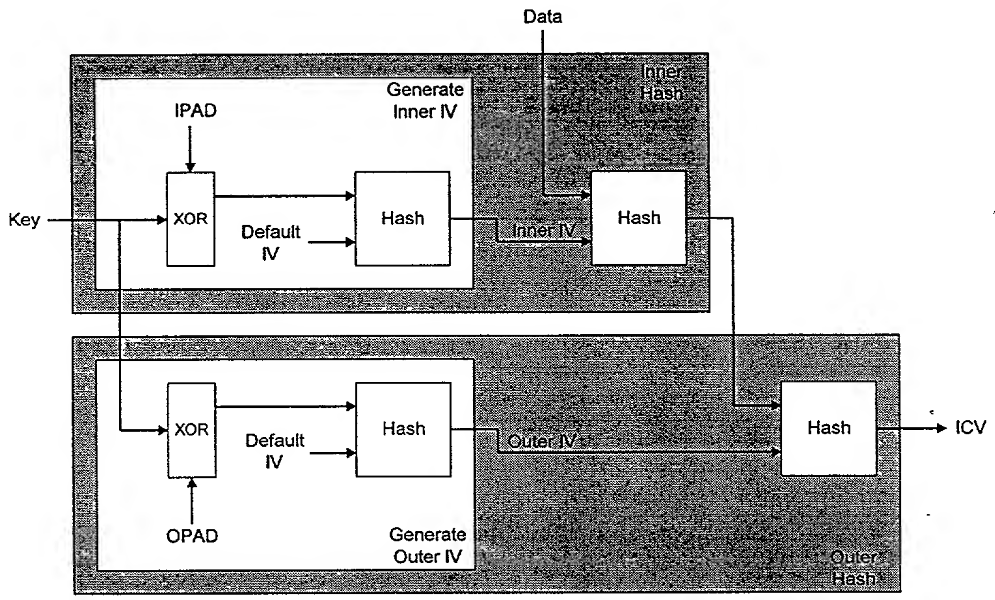


Figure 3: HMAC Operation

Table 12: HMAC Unit Commands

Command	**Description**
HMAC Complete	Perform complete HMAC operation (support key size of 160 bits for SHA-1 and 128 bits for MD5 only).
HMAC Inner IV Generation	Generate Inner IV.
HMAC Outer IV Generation	Generate Outer IV.
HMAC Update	Hash data using the provided Inner IV or intermediate result from a previous update command.

HMAC Final	Hash data, with padding appended, using the provided Inner IV or intermediate result from a previous update command, and then perform the outer hash using the provided Outer IV.
Hash Complete	Perform complete hash operation.
Hash Initialization & Update	Initialize with hash default IV and then hash data
Hash Final	Hash data, with padding appended, using the provided intermediate result from a previous update command.

Each of the HMAC units also has a set of Expected ICV Registers for storing the expected authentication value. When the Packet Processor is used, the HMAC units compare the final hash results against the expected values and report status for inbound packets.

The HMAC units are connected to the Processor Local Bus (PLB) as a slave device as well as a master device. The slave interfaces may be used to access all registers internal to the units, the 32-word HMAC 1 Input FIFO, and the 64-word HMAC 2 Input FIFO. The master interfaces provide for the more efficient DMA transfer of data to the Input FIFOs and of the authentication results (also known as Integrity Check Value or ICV) out from the Hash Registers. When the Packet Processor is enabled, the HMAC units accept inputs only from the RAM-based Controller, except for the HMAC 1 and HMAC 2 Configuration Registers.

Each HMAC unit starts processing data following a write to its HMAC Command/Status Register. While the unit is busy hashing, the Busy bit of the HMAC Command/Status Register is set to one. When processing completes, the Busy bit transitions from one to zero and a maskable critical interrupt is generated.

4.1. Address Map

The HMAC 1 and HMAC 2 registers and memory devices are memory-mapped to the PLB address bus as shown in Table 13. All of the registers, except for the HMAC 1 and HMAC 2 Configuration Registers, are also memory mapped to the Packet Processor address bus, as shown in Table 1.

Table 13: PLB Address Map for the HMAC Units

PLB Address	Device
0x20040000	HMAC 1 Configuration Register
0x20040004	HMAC 1 Length Register 0
0x20040008	HMAC 1 Length Register 1
0x2004000C	HMAC 1 Outer IV Register 0
0x20040010	HMAC 1 Outer IV Register 1
0x20040014	HMAC 1 Outer IV Register 2
0x20040018	HMAC 1 Outer IV Register 3
0x2004001C	HMAC 1 Outer IV Register 4
0x20040020	HMAC 1 Hash Register 0
0x20040024	HMAC 1 Hash Register 1
0x20040028	HMAC 1 Hash Register 2
0x2004002C	HMAC 1 Hash Register 3
0x20040030	HMAC 1 Hash Register 4
0x20040034	HMAC 1 Source Address Register

0x20040038	HMAC 1 Destination Address Register
0x2004003C	HMAC 1 Command/Status Register
0x20140000	HMAC 1 Input FIFO
0x20050000	HMAC 2 Configuration Register
0x20050004	HMAC 2 Length Register 0
0x20050008	HMAC 2 Length Register 1
0x2005000C	HMAC 2 Outer IV Register 0
0x20050010	HMAC 2 Outer IV Register 1
0x20050014	HMAC 2 Outer IV Register 2
0x20050018	HMAC 2 Outer IV Register 3
0x2005001C	HMAC 2 Outer IV Register 4
0x20050020	HMAC 2 Hash Register 0
0x20050024	HMAC 2 Hash Register 1
0x20050028	HMAC 2 Hash Register 2
0x2005002C	HMAC 2 Hash Register 3
0x20050030	HMAC 2 Hash Register 4
0x20050034	HMAC 2 Source Address Register
0x20050038	HMAC 2 Destination Address Register
0x2005003C	HMAC 2 Command/Status Register
0x20150000	HMAC 2 Input FIFO

4.2. Register Definitions

4.2.1. HMAC Configuration Register

HMAC 1 PLB Address: 0x20040000
HMAC 2 PLB Address: 0x20050000

The bit-definition of this register is shown in Table 14. This register can be updated at any time.

Table 14: HMAC Configuration Register

Bit #	Description
26:25	**Local Bus Priority.** These bits specify the PLB bus priority to be used when performing DMA transfer using the master interface. 00 specify lowest priority; 11 specify highest priority.
24	**Terminate.** This bit set commands the HMAC unit to stop immediately. This bit should be set to zero during normal operation.

4.2.2. HMAC Command/Status Register

HMAC 1 PP Address: 0x4F
HMAC 1 PLB Address: 0x2004003C

HMAC 2 PP Address: 0x6F
HMAC 2 PLB Address: 0x2005003C

The bit-definition of this register is shown in Table 15. Writing to this register starts the HMAC unit and sets the Busy bit. Writing to this register is inhibited while the Busy bit is set. The completion of processing is indicated by the transition of the Busy bit from one to zero.

Table 15: HMAC Command/Status Register

Bit #	Description
31	**Busy.** This read-only bit set indicates that the HMAC unit is busy processing data. This bit clear indicates that the HMAC unit is idle and is ready for a new command.
30	**Size Error.** This bit set indicates that the data size is not an even multiple of 64 bytes when performing the HMAC/hash update command and that the result should be discarded. This bit clear indicates that the data size is valid.
29:28	**Mode Select.** These bits specify the size of the authentication value. 00 – selects to use only the leftmost 96 bits. 01 – selects to use only the leftmost 128 bits. 1x – selects to use the entire authentication value (160 bits for SHA-1, 128 bits for MD5).
27	**HMAC inner/outer hash only.** For HMAC 1, this bit set specifies to perform the inner hash of HMAC only. For HMAC 2, this bit set specifies to perform the outer hash of HMAC only. This bit is used when performing the HMAC Final command using both HMAC 1 and HMAC 2.
26	**Output DMA AutoIncrement Disable.** This bit clear specifies to increment the destination address when using the PLB master interface to write out the results. This bit set specifies not to increment the destination address.
25	**Output DMA Enable.** This bit set specifies to write out the results using the PLB master interface. This bit clear disables the PLB master write interface.
24	**Output-To-HMAC 2 Enable / Input FIFO Early Release Enable.** This bit should be set to zero when the HMAC units are used as independent units. When the Packet Processor is enabled, this bit has different definitions for HMAC 1 and HMAC 2. The RAM-based Controller will set this bit accordingly when this register is loaded using PP address 0x09 0r 0x0A. Output-To-HMAC 2 Enable. This bit set in HMAC 1 specifies to pass the hash value to HMAC 2, using the daisy chain bus. This bit clear specifies not to pass the hash value. The number of words transferred is selected by the algorithm bits Input FIFO Early Release Enable. This bit set in HMAC 2 specifies to release the Input FIFO as soon as possible. This bit should be set to zero when processing inbound packets.
23	**Initialize Hash.** This bit set specifies to use the default initial value specified by the algorithm as the starting hash value. This bit clear specifies to use the value currently in the HMAC Hash Registers as the starting hash value.
22	**Final Block.** This bit set specifies to append padding and complete the hash operation. If the HMAC algorithm is selected, the unit will also perform the outer hash. This bit clear specifies that this is not the last block of the message and no padding or length should be appended. Note that size must be multiples of 512 bits if this bit is not set.
21:20	**Algorithm.** These bits specify the algorithm to be performed and are decoded as follow: 00 -- MD5 01 -- SHA-1 10 -- HMAC-MD5 11 -- HMAC-SHA-1
19	**Length/IPAD/OPAD Select.** When the Initialize Hash bit is set and the Final Block bit is clear, this bit is used to select between HMAC IPAD and OPAD. This bit set specifies to use the HMAC IPAD; this bit clear specifies to use the HMAC OPAD. This bit is used when performing the HMAC Inner and Outer IV

	generation commands. Otherwise, this bit is used to select the source of the message length. This bit set specifies to use the contents of the Length Registers as the length of the message; this bit clear specifies to use the Size field as the length of the message.
18	**Input DMA AutoIncrement Disable.** This bit clear specifies to increment the source address when using the PLB master interface to read in data. This bit set specifies not to increment the source address.
17	**Input DMA Enable.** This bit set specifies to read in data using the PLB master interface. This bit clear disables the PLB master read interface.
16	**Endian.** This bit set specifies that the data is in little endian format. This bit clear specifies that the data is in big endian format. Endian conversion is applied to the Input FIFO data and output hash value.
15:0	**Hash Size.** These bits specify the number of bytes to be processed. This field is initialized when this register is updated. During processing, this field is updated to reflect the number of bytes remaining to be hashed. When performing the HMAC Inner or Outer IV Generation command, this field specifies the number of bytes that are in the HMAC key.

4.2.3. HMAC Source Address Register

HMAC 1 PP Address: 0x4D
HMAC 1 PLB Address: 0x20040034

HMAC 2 PP Address: 0x6D
HMAC 2 PLB Address: 0x20050034

This 32-bit register stores the source address of the HMAC unit input data. This register is used when input DMA is enabled and the Packet Processor is disabled.

4.2.4. HMAC Destination Address Register

HMAC 1 PP Address: 0x4E
HMAC 1 PLB Address: 0x20040038

HMAC 2 PP Address: 0x6E
HMAC 2 PLB Address: 0x20050038

This 32-bit register stores the destination address of the HMAC unit output data. This register is used when output DMA is enabled.

4.2.5. HMAC Length Registers 0-1

HMAC 1 PP Addresses: 0x41, 0x42
HMAC 1 PLB Addresses: 0x20040004, 0x20040008

HMAC 2 PP Addresses: 0x61, 0x62
HMAC 2 PLB Addresses: 0x20050004, 0x20050008

These registers store the 61-bit length that specifies the total the number of bytes in the message. Length Register 0 stores the lower 32-bit of the length. Length Register 1 stores the upper 29 bits. The contents of these registers are appended to the message when the Final Block and Length Select bits of the HMAC Command/Status Register are set.

4.2.6. HMAC Outer IV Registers 0-4

HMAC 1 PP Addresses: 0x43 – 0x47
HMAC 1 PLB Addresses: 0x2004000C – 0x2004001C

HMAC 2 PP Addresses: 0x63 – 0x67
HMAC 2 PLB Addresses: 0x2005000C – 0x2005001C

When performing the HMAC Final command, these registers store the Outer IV to be used for the outer hash. When performing the HMAC Complete command, these registers store the HMAC key, which must be padded with zeroes if it is less than 128 bits (for MD5) or 160 bits (for SHA-1).

4.2.7. HMAC Hash Registers 0-4

HMAC 1 PP Addresses: 0x48 – 0x4C
HMAC 1 PLB Addresses: 0x20040020 – 0x20040030

HMAC 2 PP Addresses: 0x68 – 0x6C
HMAC 2 PLB Addresses: 0x20050020 – 0x20050030

These registers store either the Initial Value (IV) or the output hash result. These registers need to be loaded only when performing an update or final command, and in which case, either the Inner IV or intermediate result from a previous update command, as appropriate, is written to the registers. While the HMAC unit is busy processing, reading from these registers returns the hash value of the last 512-bit block.

When the HMAC unit completes processing the data, the hash value (also known as Integrity Check Value or authentication value) is stored in these registers. The final result begins with the high-order byte of Hash Register 0 and ends with the low-order byte of Hash Register 3 for MD5 and of Hash Register 4 for SHA-1. Note that byte swapping is automatically done for the final MD5 block to achieve the above result. Hash Register 4 is not used for MD5.

If output DMA is enabled, the contents of these registers are written out to the location specified by the HMAC Destination Address Register, using the PLB master write interface.

4.3. HMAC Input FIFO

HMAC 1 PLB Address: 0x20140000
HMAC 2 PLB Address: 0x20150000

The Input FIFOs of HMAC 1 and HMAC 2 may be accessed using the PLB address above. When input DMA is enabled, data is transferred into the FIFOs using the PLB master interfaces. When the Packet Processor is used, the Input FIFOs accepts data only from the RAM-based Controller, through the use of the WRITE_DATA instruction. When performing the HMAC Inner or Outer IV Generation command, the Input FIFO is used to store the HMAC key.